SCHEDULING WITH RESTRICTED MACHINE AVAILABILITY AND DUE DATE

By

SURYA DANUSAPUTRO LIMAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1991

I would like to dedicate this work to my beloved parents: Anwar Liman and Tin Suwargo, especially to my father who passed away on March 16, 1991. I owe everything that I am to them and I only wish that he could be here to share it with me. I would also like to dedicate it to my loving wife, Hong Li.

ACKNOWLEDGMENTS

I would like to extend my sincerest gratitude and appreciation to Dr. Chung-Yee Lee, chairman of my supervisory committee, without whose guidance and assistance this work could not have been completed. I would also like to thank Drs. Elzinga, Erenguc, and Martin-Vega for making themselves available to serve on my supervisory committee and for helping me in the preparation of this dissertation. Thanks is also due to Dr. Sivazlian who has helped me in many ways throughout my years as a graduate student. And last but not least, my utmost appreciation is due to my dear wife, Hong Li, who has stood by me through all these years and who has faith in me that I can do what I set out to do: get the Doctor of Philosophy degree and get a job.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SCHEDULING WITH RESTRICTED MACHINE AVAILABILITY AND DUE DATE

By

Surya Danusaputro Liman

August 1991

Chairman: Dr. Chung-Yee Lee
Major Department: Industrial and Systems Engineering

We consider scheduling problems with restricted machine availability and due date. For problems with restricted machine availability, we investigate the minimization of the sum of job completion times under the following three cases: (1) machines are not all simultaneously available (rolling horizon problem), (2) machines are only available for a specified period of time, and (3) machines are taken out of the production line for a preventive maintenance once during the associated production cycle. For the first problem, we prove that the Shortest Processing Time (SPT) algorithm yields an optimal solution. For the second problem, we show it to be NP-Complete, provide a dynamic programming algorithm, propose a new heuristic which has a tight worst case error bound of 0.5, and empirically show that the overall average error is not more than 0.03. For the third problem, we show that it is NP-

Complete, prove that a slight modification of the SPT algorithm has a tight worst case error bound of less than 2/7, and propose a new dynamic programming algorithm for the two-machine case. For the due date problem, we propose a new heuristic for the common due date problem of minimizing the sum of job earliness and tardiness. The heuristic is then proven to have a tight worst case error bound of 0.5. However, we empirically show that the overall average error is less than 0.03. We also report on simulation results for the rolling horizon problem. In particular, we investigate the performances of two different types of rolling horizon release policies under different job and shop conditions. The two rolling horizon release policies studied are: (1) Rolling Horizon with Priority for Previously Released Jobs and (2) Rolling Horizon without Priority for Previously Released Jobs.

CHAPTER 1
INTRODUCTION

Scheduling is becoming increasingly important in today's manufacturing industries. It no longer takes the back seat it once occupied in the manufacturing processes. A number of years ago, it was sufficient to be able to make products that were of high quality and low priced. However, since the advent of the Just-In-Time (JIT) philosophy -- first conceptualized, preached, practiced, and successfully implemented by the Toyota manufacturing facilities in Japan -- an equally strong emphasis has been placed on product deliverability.

One of the simple objectives of JIT is to be able to make the right product in the right amount at the right time. Today's consumers not only require low cost, high quality products but also rely on their timely deliveries. Many companies have started to implement the JIT manufacturing philosophy, which requires their vendors to deliver raw materials or subcontracted products on time for the subsequent production operations.

One of the underlying approaches of the JIT philosophy is that better control of the manufacturing system is attained through the reduction of product lot size. The ultimate objective is being able to make products in single unit lot sizes. With better control comes the ability to deliver more products in time for consumption or subsequent production processes. An additional advantage that is

afforded by better control on the manufacturing system is tighter control of the quality of the intermediate production processes, which will eventually lead to higher final product quality. High quality products result in reduced scrap and rework, which in turn lower the associated production costs.

Such control can be achieved by finding better production schedules. This is a direct result of the frequent product changeovers resulting from the smaller lot size. In light of the ever increasing awareness of the importance of JIT and the sheer complexity of today's manufacturing processes, new and innovative scheduling techniques are bound to have more and more applications in industries.

A number of noteworthy books, such as Conway et al. (1967), Baker (1974), Coffman (1976), Rinnooy Kan (1976), Lenstra (1977), and French (1982), have been written on the subject of scheduling. Before we go any further, we would like to first formally define the term "scheduling."

The term "scheduling," which has its origin in the manufacturing context but has other important applications in areas such as vehicle routing and crew scheduling, is defined by Dempster et al. (1982, p.3) as

> the art of assigning [scarce] resources to tasks in order to insure termination of these tasks in a reasonable amount of time.

Scheduling is an art since, as we will see, most of the real life scheduling problems cannot be efficiently and analytically

solved. Hence practical solutions often depend on some form of human ingenuity, instinct, experience, and judgement. By assignment, we mean a certain ordering or sequencing that is to be the eventual outcome of this venture. Resources are defined in the classical sense to be machines or processors with limited capacities. Tasks are interpreted as products, jobs, or anything that requires the use of the above scarce resources. The reasonability of the amount of time is a measure of goodness of the assignment of these resources to the tasks. That is, it is a measure of performance of the scheduling policies.

Scheduling differs in certain aspects from "sequencing" in that the latter deals only with the specific ordering of products, while the former gives a more complete description as to when to start a particular production run. The two terms are often used interchangeably since in most situations, once the sequence is determined, we can compute its corresponding schedule.

There are numerous performance measures that can be adopted. However, they can be grouped into two distinct categories: (1) regular and (2) nonregular performance criteria (Conway et al., 1967; Baker, 1974). Conway et al. (1967, p. 12) define a regular measure as

> a value to be minimized that can be expressed as a function of job completion times, ... , and which increases only if at least one of the completion times increases. Measures such as the mean or the maximum of flow times, lateness, and tardiness fall into this category while measures such as mean earliness and weighted sum of earliness and tardiness are nonregular measures.

As to which type of performance measure is more frequently used, it is usually related to one company's perception as its most important goals. For example, if rapid product turnover is to be achieved, an obvious choice would be to minimize the sum of job flow times defined as the total length of time the jobs actually spend in the system. On the other hand, if the company decides to minimize customer complaints about late deliveries, the obvious choice would be the minimization of some penalties associated with the job due date. The importance of due date related measures is described by Conway et. al (1967, p. 30) as

> the measure that arouses the most interest in those who face practical problems of sequencing is the satisfaction of preassigned due dates ... the ability to fulfill delivery promises on time undoubtedly dominates these other considerations.

Due dates provide a vital role in the production planning cycles, driving issues such as planned order release and material requirement planning (see Figure 1-1). For a survey of due date assignment procedures, please see Seidman and Smith (1981).

In some instances, due dates are specified by the customers (exogenous due dates), while in others due dates can be internally determined by the manufacturers (endogenous due dates). In some industrial settings, due dates can be treated as decision variables within the boundary of the scheduling problem. That is, they are negotiable with the customers.

After analyzing a number of real industrial problems, we come to the conclusion that most companies regard maximizing throughputs

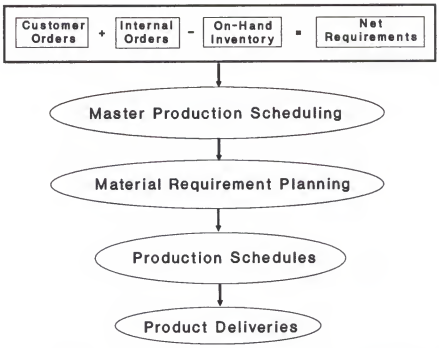


Figure 1-1. Production Planning Cycle.

and meeting customer due dates as the two most important goals. We will therefore investigate both the minimization of sum of job flow times and the minimization of due date related penalty functions.

For the minimization of mean flow time, we look at the effect of the Shortest Processing Time (SPT) algorithm on restricted machine availability problems. That is, given that we schedule the jobs using the SPT algorithm, we want to study the performance of the algorithm when the machines are not always available. In particular, we investigate the following three problems: (1) machines are not all simultaneously available, (2) machines are available only for a specified period of time, and (3) machines are taken out of the production line for preventive maintenance once during the associated

production cycle. These circumstances make the problems more realistic since we relax the classical restrictive assumptions of simultaneous and continuous machine availability. We will investigate both the single machine and multiple parallel machines systems.

For the due date related measures, we will study the minimization of the sum of absolute deviations of job completion times about a common due date (also commonly known as the Earliness-Tardiness [ET] problem), which is a nonregular measure of performance. This measure is in line with the JIT concept in that it discourages jobs that are completed either too early or too late. In particular, we look at the special case where there is only one due date for the entire set of products (a common due date).

The manufacturing facility is assumed to be that of a shop floor. The most general shop scenario considered in the literature is the job shop environment. This type of scheduling problems has been proven to be one of the hardest problems to solve in the sense that it is extremely difficult if not impossible to obtain optimal solutions. That is, most of the job shop problems are classified as NP-Complete problems in the strong sense (see Karp, 1972, for the first seminal paper on NP-Completeness of combinatorial problems and Garey and Johnson, 1979, for the theory of NP-Completeness).

The problems that will be subsequently discussed are assumed to exist in such a job shop environment. It is therefore appropriate to provide a common ground and concrete foundation for this environment

before anything else. In the next section, we will state the general conditions and assumptions that prevail in a job shop as described by Conway et al. (1967).

The Job Shop

A general job shop is a set of machines that are required for the production or processing of jobs. It is assumed to consist of m work centers, each of which may contain one or more identical machines that run in parallel. Each job has a predetermined, required sequence of operations to be performed in these work centers. This sequence of operations may vary between jobs and may require processing by more than one of the work centers. There are precedence constraints that dictate the ordering of operations for these jobs. These constraints are sometimes called the job technological ordering.

The availability of the jobs dictates the type of problem studied. If all jobs are initially available prior to the time the scheduling is made, the problem is known as a static scheduling problem. However, if the jobs are made available over time (i.e., jobs have different ready or release times), the problem is known as a dynamic scheduling problem. These jobs may require processing by all or a subset of these m work centers.

The treatment of the general job shop has been shown to be extremely difficult. Hence, for the most part, we will be looking at the following special case: one work center job shop problem. That is, a job shop that consists of only one work center. In some cases,

we will assume that the work center contains one machine, while in others we will allow more than one identical machine. For each problem we discuss, we will specifically identify the environment of our problem formulation.

By studying single work center systems in detail, we hope to attain greater understanding of the more complex systems. With this understanding, we may be able to predict the behavior of the more complicated job shop system that we initially describe. Such understanding may also lead to the development of simple scheduling algorithms that can be extended to the systems with more than one work center.

Excellent introductions and surveys of scheduling policies, theories, and practices can be found in Day and Hottenstein (1970), Panwalkar and Iskander (1977), Graves (1981), Dempster et al. (1982), Lawler et al. (1982), Lenstra and Rinnooy Kan (1984), Blazewicz et al. (1988), Kawaguchi and Kyan (1988), Rodamer and White Jr. (1988), Buxey (1989), and Lawler et al. (1989), among others. Lageweg et al. (1981a and b) provide excellent reviews on the complexities of scheduling problems.

Notation

We will now define the following notation, which will be consistently used through out the dissertation. Additional notation will be defined as needed.

n = number of jobs

m = number of machines

D_j = due date of job j

P_j = processing time of job j

C_j = completion time of job j

E_j = max (0, D_j-C_j) = earliness of job j

T_j = max (0, C_j-D_j) = tardiness of job j

Outline

The main objective of this dissertation is to advance the theoretical understanding of the scheduling process. We also intend to point out some applications of the problems that we investigate. Even though a vast number of scheduling problems have been looked at and even solved, the number of unsolved problems is still staggering. Clearly, it is not our intention to solve the entire scheduling problem in existence, an attempt that would definitely prove to be futile. With the further understanding of these underlying theoretical values, we ultimately hope to be able to afford better controls of the manufacturing processes and thus be able to achieve more on time deliveries.

Unlike the regular practice of gathering all the literature surveys into one chapter, we will present the review within the body of the chapters since the topics discussed in each chapter are relatively independent. The remainder of the dissertation is presented as follows.

The following two chapters deal specifically with the problems posed above: scheduling with restricted machine availability and scheduling with nonregular measure of performance. At the beginning

of each chapter, we will first provide some motivations for the study of the particular problems. Some basic notation and terminologies used in describing the problem and its solutions are presented next. We will then review relevant literature and algorithms that exist for the problems studied. We will then present our theoretical results and extensions, along with some practical applications for the problems.

In Chapter 4, we will investigate the performance of one of the algorithms presented in chapter 1 and report on the simulation results obtained. In the last chapter, we will summarize all the results, derive some conclusions, and define future research directions.

CHAPTER 2
SCHEDULING WITH RESTRICTED MACHINE AVAILABILITY

Introduction and Motivations

In this chapter, we will study the static (all jobs are initially available) scheduling problem of minimizing the sum of job flow times with restricted machine availability (i.e., machines having some capacity constraints). The flow time of a job is defined as the total elapsed time the job spends in the system (from arrival to completion). This measure of performance has been shown to be equivalent to a measure that maximizes throughput and minimizes work-in-process inventory (see for example Baker, 1974, pp. 14-17).

We assume that the system consists of one work center, which may contain one or more identical machines. We also assume that preemption is not allowed (i.e., once a job has started its processing, it cannot be interrupted until it has completed its processing). In some instances, this assumption can be viewed as a preempt-repeat situation (i.e., if a job's processing has to be interrupted, the partial processing that has been done is completely lost and has to be repeated).

Most of the existing scheduling literature on the problem of minimizing the sum of flow times assumes that machines are simultaneously available and continuously operative until the last job is completed. Under these restrictive assumptions, it has been

shown that the Shortest Processing Time (SPT) algorithm produces an optimal schedule. The SPT algorithm (see for example Smith, 1956; Conway et al., 1967; Baker, 1974) iteratively assigns the job with the smallest processing time among all yet-to-be-scheduled jobs to the earliest available machine (machine whose last job's completion time is the smallest).

We will relax these classical restrictions on machine availability. In particular, we will study three instances of the relaxation: (1) rolling horizon with nonsimultaneous machine availability, (2) machines with prior commitments, and (3) preventive maintenance problems. We will investigate the performance of the SPT algorithm under these relaxations to see if it still yields optimal schedules. If it does not, then we will investigate if the problem is NP-Complete. If it is NP-Complete, we will then simultaneously propose a pseudo-polynomial dynamic programming algorithm to optimally solve the problem and look at the worst case performance of the SPT algorithm. We will also provide some empirical results for the average performance of the algorithm.

Definitions and Overview of Terminologies

Throughout this chapter, we assume that jobs are numbered in nondecreasing order of processing times (i.e., $p_1 \leq p_2 \leq \cdots \leq p_n$). In addition to the notations mentioned in the previous chapter, we define the following additional notations (see Figure 2-1):

U_i = time machine i becomes available

R_i = start time of maintenance on machine i

L = length of time of maintenance

B_i = set of jobs processed on machine i before maintenance

A_i = set of jobs processed on machine i after maintenance

|X| = cardinality of set X

F(S) = sum of job flow times of schedule S

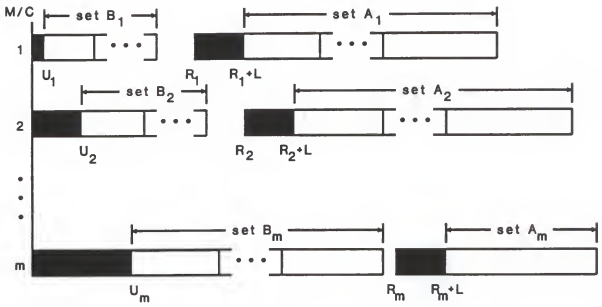


Figure 2-1. A Feasible Schedule to the Problem.

Literature Review

Literature that deals with the rolling horizon with nonsimultaneous machine availability problem is very limited. In fact, as far as we know only Lee (1991) has studied the problem. He looks at the problem of minimizing the maximum job completion time (makespan) when machines are not simultaneously available. He shows that the LPT algorithm, which assigns the longest unassigned job to the minimally loaded machine, has a worst case error bound of 0.5.

He also shows that this bound is tight (i.e., one can create a particular instance of the problem that attains this error bound). He then provides an alternative algorithm called Modified LPT algorithm, which is proven to have a better worst case error bound of 1/3.

Kao and Elsayed (1989) look at the two-machine capacitated problem whereby they assume Machine 1 to be continuously available but Machine 2 is only available for a specified period of time. In their paper, they formulate the problem as a zero-one integer program and provide a heuristic to solve the problem. However, they do not prove the problem to be NP-Complete. The underlying idea of their heuristic is to first obtain an initial optimal schedule for the problem without the capacity constraint. They then use some interchange and transfer procedures to reduce the makespan of the capacitated machine. They also provide a worst case error bound for the heuristic that depends on the number of interchanges and the reduction in the makespan of the machine with capacity.

Literature on predetermined preventive maintenance is scarce. Most researchers investigate the problem as a stochastic process whereby the preventive maintenance or machine breakdown times are assumed to be random variables that follow some distribution function. In the context of the deterministic problem, the first paper was written by Adiri et al. (1989). They provide a rather lengthy and incomplete proof of the NP-Completeness of the one-machine problem (the complete proof is given in another technical

report). They then study the performance of the SPT algorithm and show that the schedule obtained using the SPT algorithm has a tight worst case error bound of 1/4, which we will later show to be erroneous.

Rolling Horizon with Nonsimultaneous Machine Availability Problem

In this section, we will look at the performance of the SPT algorithm under the rolling horizon job release policy in minimizing the sum of job flow times. A job release policy is defined as a procedure that releases jobs into the shop floor depending on some criteria such as shop congestion level or every fixed time interval (horizon). For example, given that jobs continuously arrive over time, the supervisor may decide to accumulate new jobs and only release them into the shop floor at the beginning of every shift (horizon). We call this type of job release policy the "rolling horizon" job release policy in the sense that new jobs are "rolled" over from one horizon to the next.

The use of this release policy in effect reduces a dynamic scheduling problem to a static problem. Hence, instead of looking at the continuous arrival of jobs, we can look at the beginning of each horizon with a number of jobs available to be scheduled within that horizon. From our experience, many companies adopt similar procedures due to their simplicity. We will report more on job release policies in Chapter 4 when we compare the performance of two types of rolling horizon release policies.

We will now study one of the two types of the rolling horizon policies under the assumption that previously released jobs have to be processed before new jobs. In this case, the machines may not be simultaneously available at the beginning of the horizon due to the fact that there may be some jobs left over from the previous horizons that may still require additional operations. In effect, we only allow a machine to begin the processing of a new job if there is no more job from the previous horizons.

We assume that the single work center contains m parallel identical machines and that machine i becomes available at time U_i. Furthermore, we also assume that the machines are numbered such that $U_1 \leq U_2 \leq \cdots \leq U_m$ and once they are available they will be continuously available thereafter.

As has been mentioned, Lee (1991) has looked at a similar problem with a different objective function. However, for the case of the minimization of the sum of flow times, we will prove that the SPT algorithm will yield an optimal sequence.

Theorem 2.1: For the nonsimultaneous machine availability problem of minimizing the sum of job flow times, the SPT algorithm will yield an optimal sequence.

The proof of Theorem 2.1, uses the following lemmas.

Lemma 2.1: There exists an optimal sequence whereby jobs assigned to machine i, $i = 1,\ldots,m$, are arranged in nondecreasing order of processing times (SPT order).

Proof: Proof is straight forward using a pairwise interchange argument.

Lemma 2.2: For any two machines, which for simplicity we denote by Machines 1 and 2, suppose that Machines 1 and 2 become available at times U_1 and U_2 respectively with $U_1 \leq U_2$, then there exists an optimal solution such that the number of jobs assigned to Machine 1 after U_1 is not less than the number of jobs assigned to Machine 2 after U_2.

Proof: By Lemma 2.1, we only need to be concerned with solutions such that jobs in any machine are arranged in SPT order. Consider a solution such that $U_1 \leq U_2$ and the number of jobs assigned to Machine 1 (Machine 2) after U_1 (U_2 respectively) in the solution is denoted by N_1 (N_2 resp.). If $N_1 < N_2$, let the smallest processing time of jobs in Machine 2 be denoted by p_j (please note that job j is the first job after U_2). In this case, if we remove job j from Machine 2 and insert into Machine 1 immediately after U_1, then the net change in the total flow times is

$$\begin{aligned}\Delta_{net} &= ((U_1+p_j) + N_1 p_j) - ((U_2+p_j) + (N_2-1)p_j) \\ &= (N_1-(N_2-1))p_j + (U_1-U_2) \\ &\leq 0 \quad \text{since } N_1 < N_2 \text{ and } U_1 \leq U_2.\end{aligned}$$

Hence, we must have $N_1 \geq N_2$. Q.E.D.

We are now in the position to prove Theorem 2.1.

Proof of Theorem 2.1: Without loss of generality, assume that U_1 is the earliest time a machine becomes available. Note that in an optimal solution, Lemma 2.1 implies that job 1 is assigned after U_i for some i and Lemma 2.2 implies that $N_1 \geq N_i$. Suppose that $i > 1$, let job j be the first job that is assigned to Machine 1 immediately after U_1. Interchanging job 1 and job j, the net change in total flow times is

$$\Delta_{net} = (N_i - N_1)(p_j - p_1)$$
$$\leq 0 \quad \text{since } N_i \leq N_1 \text{ and } p_j \geq p_1.$$

We conclude that p_1 must be assigned to the earliest available machine. By repeating the same argument, we can see that the SPT algorithm will result in an optimal sequence. Q.E.D.

Machines with Prior Commitments Problem

In this section we present some results on the scheduling problem of machines with prior commitments. In particular, we look at the two-parallel machines scheduling problem with the objective of minimizing the sum of job flow times. We assume that both machines are all initially available (i.e., $U_1 = U_2 = 0$). However, instead of allowing them to be continuously available as is often assumed in the literature, we will only allow one of the machines to be continuously available, while the other is available only for a specified period of time after which it can no longer process any job. Such circumstances may arise when that machine has prior commitments to process other jobs.

We will call this problem the capacitated sum of job flow times (CSFT) problem. Without loss of generality, we will assume that Machine 2 has a capacity constraint and can only process jobs up to time R_2, while Machine 1 is continuously available.

The motivation behind the study of this particular problem lies in the curiosity as to the performance of the SPT algorithm in cases of unforeseen circumstances. The SPT algorithm yields optimal schedules in the absence of any disruptions to the schedules. However, a situation may arise when one machine (say, Machine 2) breaks down (at time R_2) and thus can no longer process the jobs assigned to it. In this case, the job whose processing has already been started on Machine 2 has to be repeated from the beginning (preemptive resume situation). This job and the remaining ones that are scheduled to be processed on that machine will subsequently join the yet-to-be-processed jobs on Machine 1. These jobs are then processed according to the SPT sequence. We want to analyze the robustness of the SPT algorithm if this situation should arise.

As was mentioned, the CSFT problem has been studied by Kao and Elsayed (1989). In that paper they provide a heuristic algorithm without providing any justification as to why a polynomial time algorithm does not exist. In the following sections, we will first prove that the CSFT problem is indeed NP-Complete and then provide a pseudo-polynomial dynamic programming algorithm to solve the problem. We will also propose a heuristic that has a tight worst case error bound of 0.5.

NP-Completeness of the CSFT Problem

In this section, we will first prove that the CSFT problem is NP-Complete (please refer to Garey and Johnson, 1979, for the theory of NP-Completeness). We will provide the proof by transforming the even-odd partition problem, which has been shown to be NP-Complete (Garey et al., 1988), in polynomial time into the CSFT problem.

The even-odd partition problem:

Given $n \in Z^+$ and a set $X = \{x_1, x_2, \ldots, x_{2n}\}$ of positive integers, where $x_i < x_{i+1}$ for $1 \le i < 2n$, does there exist a partition of X into subsets X_1 and X_2 such that $\Sigma_{x \in X_1} x = \Sigma_{x \in X_2} x$ and such that for each i, $1 \le i \le n$, X_1 (and hence X_2) contains exactly one of $\{x_{2i-1}, x_{2i}\}$?

Let $Z = (\sum_{i=1}^{2n} x_i)/2$. The corresponding CSFT problem can be constructed as follows:

Number of jobs: $2n+1$

Processing times: $p_i = x_i, \quad i=1,\ldots,2n$

$$p_{2n+1} = \sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i$$

Capacity of Machine 2: $R_2 = Z$

Sum of flow times: $y = 2\sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i + R_2$

Question: *Does there exist a nonpreemptive schedule S^* such that the completion time of the last job in Machine 2 is not greater than R_2*

and the sum of flow times of all jobs in S^*, denoted by $F(S^*)$, is no more than y?

Remark: Note that in the above instance, $0 < p_1 < p_2 < \cdots < p_{2n+1}$ and that it can be created in polynomial time.

Given any schedule, we will now define the following sets (see Figure 2-2). Let B_i be the set of jobs assigned to machine i (i=1,2) with completion times not greater than R_2 and A_1 be the set of jobs assigned to Machine 1 other than jobs in set B_1. Hence $B_1 \cup A_1$ is the set of all jobs assigned to Machine 1.

Lemma 2.3: If there exists a solution to the even-odd partition problem, then there exists a schedule S^* for the CSFT problem with the sum of job flow times $F(S^*) = y$.

Proof: We can construct a schedule S^* of the jobs as shown in Figure 2-2, where the jobs in B_1 (B_2) are representative of the elements of set X_1 (X_2 respectively) and all the jobs in B_1 and B_2 are in nondecreasing order of their processing times.

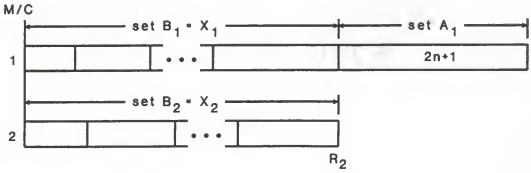


Figure 2-2. Solution to CSFT.

Since the solution to the even-odd partition exists, we have

$$\Sigma_{x\in X_1} x = \Sigma_{x\in X_2} x = Z = \Sigma_{i\in B_1} p_i = \Sigma_{i\in B_2} p_i = R_2$$

Hence

$$\begin{aligned} F(S^*) &= \sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) + (R_2 + p_{2n+1}) \\ &= \sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) \\ &\quad + (R_2 + \sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i) \\ &= 2\sum_{i=1}^{n} (n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i + R_2 \\ &= y \end{aligned}$$

Q.E.D.

Lemma 2.4: Let S^* be a feasible schedule to the CSFT problem with $F(S^*) \le y$, then in S^*, the following must be true:

(1) job 2n+1 is processed last in Machine 1,

(2) $\Sigma_{j\in B_2} p_j = R_2$ and exactly one of jobs 2i-1 or 2i is in B_1 and the other is in B_2 for $i = 1,\ldots,n$.

Proof: To prove part (1), we will show that if job 2n+1 is not processed last in Machine 1, then $F(S^*) > y$. It can be easily checked that $p_{2n+1} > R_2$. Hence job 2n+1 cannot be processed by Machine 2 and must be processed by Machine 1. Now, suppose job 2n+1 is not processed last in Machine 1. This implies that there is another job, say job j, that is processed after job 2n+1. Then

$$\begin{aligned} F(S^*) &\ge p_{2n+1} + (p_{2n+1} + p_j) \\ &= 2p_{2n+1} + p_j \end{aligned}$$

$$= 2\left(\sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i\right) + p_j$$

Since $R_2 = Z = (\Sigma_{x\in X} x)/2 = (\sum_{i=1}^{2n} p_i)/2$, we have

$$F(S^*) > 2\sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i + R_2$$

$$= y$$

Hence, by contradiction, job 2n+1 has to be processed last in Machine 1. From now on we will only consider the first 2n jobs.

To prove part (2), let $\Sigma_{j\in B_2} p_j = R_2 - \delta$, $\delta \geq 0$ (see Figure 2-3). We want to show that $\delta = 0$.

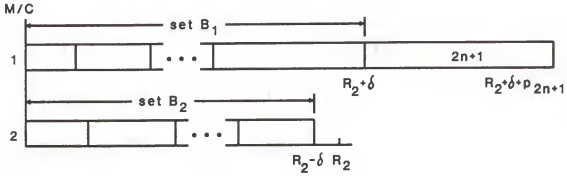


Figure 2-3. Schedules S'.

It is well known that for 2n-job two-parallel machines problem without capacity constraint, the optimal sequencing policy, SPT, will yield the minimum sum of flow times as follows

$$F_{min} = \sum_{i=1}^{n}(n-i+1)(p_{2i-1}+p_{2i})$$

$$= \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i})$$

Let F_{2n} be the sum of flow times for the first 2n jobs in schedule S^*. Then $F_{2n} \geq F_{min}$ since F_{min} is a lower bound for the first 2n jobs problem. Also from Figure 2-3 we have

$$\begin{aligned} F(S^*) &= F_{2n} + (R_2 + \delta + p_{2n+1}) \\ &\geq F_{min} + (R_2 + \delta + p_{2n+1}) \\ &= \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + (R_2+\delta+\sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + \sum_{i=1}^{2n} p_i) \\ &= y + \delta \end{aligned}$$

If $\delta > 0$, then $F(S^*) > y$, contradicting the hypothesis. Hence $\delta = 0$ and $F(S^*) = y$. This implies that $\Sigma_{j\in B_2} p_j = R_2$ and $F_{2n} = F_{min}$.

By the classical SPT argument for minimizing the sum of flow times (Baker, 1974), any 2n-job schedule that attains F_{min} must be scheduled by consecutively assigning the two smallest unassigned jobs to Machines 1 and 2 at the same relative position. Hence F_{2n} must be assigned in this way. Q.E.D.

Remark: Part (2) in Lemma 2.4 implies that there exists an even-odd partition. Combining Lemma 2.4 with Lemma 2.3 we have shown the following theorem.

Theorem 2.2: The capacitated sum of flow times problem is NP-Complete.

Dynamic Programming Algorithm

We will now provide a pseudo-polynomial dynamic programming algorithm to solve the CSFT problem, which together with the the NP-Complete proof implies that the problem is NP-Complete in the ordinary sense. A pseudo-polynomial algorithm is an algorithm whose complexity depends on a particular instance of the problem, such as the sum of job processing times.

Algorithm 2.1

Let $f(j,t)$ = minimum sum of job flow times if we have assigned jobs $1,\ldots,j$ and the completion time of the last job in Machine 2 is t.

Boundary condition:

$$f(0,0) = 0$$

$$f(0,t) = \infty \text{ for } t \neq 0 \tag{2.1}$$

$$f(j,t) = \infty \text{ for } t < 0 \text{ and } j > 0$$

Recursive Relation:

$$f(j,t) = \min \begin{cases} f(j-1,t) + (s_j - t) & (2.2) \\ f(j-1,t-p_j) + t & (2.3) \end{cases}$$

where $j = 1,\ldots,n$, $t = 0,\ldots,R_2$, and $s_j = \Sigma_{i \leq j} p_i$

Solution:

$$F(S^*) = \min \{f(n,t) : t = 0,\ldots,R_2\} \tag{2.4}$$

Complexity:

Since $j = 1,\ldots,n$ and $t = 0,\ldots,R_2$, the complexity of Algorithm 2.1 is $O(nR_2)$.

We now give the justification for Algorithm 2.1. Consider the case when job j is the next job to be scheduled and that the flow time of the last job in Machine 2 is $t \leq R_2$. Job j is either assigned to Machine 1 or Machine 2 depending on which of the two will result in the minimum sum of job flow times. If job j is assigned to Machine 1 (see Figure 2-4a), then the additional flow time will be the flow time of job j ($= \Sigma_{i \leq j} p_i - t = s_j - t$), which added to the sum of job flow times prior to assigning job j ($= f(j-1,t)$) resulted in equation (2.2). On the other hand, if job j is assigned to Machine 2 (see Figure 2-4b), the additional flow time will be the flow time of job j ($= t$), which added to the sum of job flow times prior to assigning job j ($= f(j-1,t-p_j)$) resulted in equation (2.3).

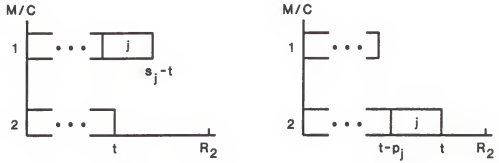


Figure 2-4. Placements of Job j.
a) j is assigned to Machine 1 b) j is assigned to Machine 2

Heuristic

Since the problem is NP-Complete, in many instances an efficient heuristic is justifiable. Kao and Elsayed (1989) propose a heuristic in which they first obtain an optimal schedule for the problem without the capacity constraint and then use interchange

procedures to reduce the makespan of the machine with capacity until it is no longer violated. It is interesting to study the performance of the rather straightforward SPT algorithm for the capacitated machine problem. We will now provide an algorithm based on the SPT algorithm with a slight modification to take into consideration the capacity of Machine 2. This approach is different from the one proposed by Kao and Elsayed (1989). We will show that the heuristic has a tight worst case error bound of 0.5. The heuristic is formally stated as follows:

Algorithm 2.2

1. Apply the SPT algorithm, two jobs at a time, beginning with Job 1 assigned to Machine 1 and Job 2 assigned to Machine 2, until such time as no other job can be assigned to Machine 2 without violating the capacity.
2. Assign the remaining jobs to the first machine.

Lemma 2.5: If the sequence obtained by Algorithm 2.2 satisfies any one of the following two properties, then it is optimal.

(1). The remaining number of jobs in step 2 is either 0 or 1.

(2). The sum of processing times of jobs assigned to Machine 2 is exactly equal to R_2.

Proof: The proof of part (1) is straightforward. If the remaining number of jobs in step 2 is either 0 or 1, this implies that the capacity of Machine 2 does not constrain the schedule. Hence it is optimal.

We will use induction to prove part (2). Let the number of jobs in sets B_1 (or equivalently B_2) be denoted by h and A_1 be denoted by j. We assume that the sum of job processing times already assigned to Machine 2 is equal to R_2. From part (1), we know that if j = 0 or 1, the sequence obtained is optimal. Suppose that the partial schedule S_k when j = k is optimal. Let S_{k+1} be the schedule with j = k+1 and let the optimal sum of flow times of schedules S_k and S_{k+1} be denoted by $F(S_k)$ and $F(S_{k+1})$ respectively. Then

$$F(S_{k+1}) \geq F(S_k) + (\sum_{i=1}^{2h+k+1} p_i - R_2)$$

But the flow time of job in k+1 position of set A_1 by applying Algorithm 2.2 is

$$\sum_{i=1}^{2h+k+1} p_i - R_2$$

Hence the total flow times of Algorithm 2.2 for j = k+1 is given by

$$F(S_k) + (\sum_{i=1}^{2h+k+1} p_i - R_2)$$

which is not greater than the optimal value $F(S_{k+1})$. Therefore, by induction, if part (2) of Lemma 2.5 is satisfied, the sequence obtained is optimal. Q.E.D.

We will now show that Algorithm 2.2 has a worst case error bound of 0.5 and that this bound is tight.

Theorem 2.3: Let S be the sequence obtained by applying Algorithm 2.2 for the problem considered. Then $F(S) \leq 1.5\ F(S^*)$, where F(S)

and $F(S^*)$ are the total flow times for schedule S and optimal schedule S^* respectively. Furthermore, this error bound is tight.

Proof: Let the cardinality of set A_1 ($|A_1|$) be denoted by z. From part (1) of Lemma 2.5, if $z \leq 1$, then S is optimal. Hence we will only consider $z \geq 2$. Let the first and second jobs in set A_1 of schedule S be denoted by x_1 and x_2 with processing times r_1 and r_2 respectively. Let the last jobs in set B_1 and B_2 be denoted by y_1 and y_2 with completion times t_1 and t_2 respectively. Note that $t_1 \leq t_2$ and $(t_2-t_1) < r_1$ for otherwise job x_1 would have been assigned to Machine 2 (see Figure 2-5).

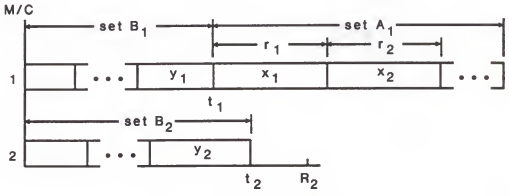


Figure 2-5. Schedule S.

Since the problem has been proven to be NP-Complete, we cannot find the true optimal value in polynomial time. Hence we will find a lower bound of the optimal solution to the problem in order to obtain a worst case error bound. In order to do this, we will construct a new problem P' with capacity R_2' such that $R_2' = t_2 + r_2$. Note that $R_2' \geq R_2$ and that any feasible solution to P will also be a feasible

solution to P'. Hence the optimal value of P' will be a lower bound for that of P.

By part (2) of Lemma 2.5, if we move job x_2 of schedule S to Machine 2 and have it completed exactly at R_2' (see Figure 2-6) we will obtain an optimal schedule S' to P'. Let S^* be the optimal schedule for problem P. Hence F(S') will be a lower bound on the optimal value to P, $F(S^*)$.

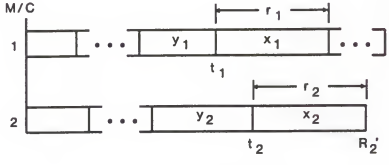


Figure 2-6. Schedule S'.

From Figures 2-5 and 2-6, we obtain

$$F(S)-F(S^*) \leq F(S)-F(S') = (t_1+r_1-t_2) + (z-2)r_2$$
$$\leq r_1 + (z-2)r_2 \qquad (2.5)$$

Furthermore, for schedule S' (see Figure 2-6), since jobs on Machine 1 that were assigned after job x_1 have processing times $\geq r_2$, we have

$$F(S') \geq (z-1)t_1 + ((z-1)r_1 + (z-2)r_2 + (z-3)r_2$$
$$+ \cdots + r_2) + (t_2+r_2)$$
$$= (z-1)t_1 + (z-1)r_1 + ((z-1)(z-2)/2)r_2 + (t_2+r_2)$$
$$\geq (z-1)r_1 + ((z-1)(z-2)/2)r_2 + r_2 \qquad (2.6)$$

Hence, using equations (2.5) and (2.6), we obtain

$$\begin{aligned}\epsilon &= (F(S)-F(S^*))/F(S^*) \\ &\le (F(S)-F(S'))/F(S') \\ &\le (r_1+(z-2)r_2)/[(z-1)r_1+((z-1)(z-2)/2)r_2+r_2] \\ &= (2r_1+2(z-2)r_2)/(2(z-1)r_1+(z-1)(z-2)r_2+2r_2) \qquad (2.7)\end{aligned}$$

From (2.7), we know that for $z = 2$

$$\epsilon = 2r_1/(2r_1+2r_2) < 0.5 \quad \text{since } r_1 \le r_2.$$

For $z \ge 3$, let $\epsilon = (a+b)/(c+d)$ where

$a = 2r_1$

$b = 2(z-2)r_2$

$c = 2(z-1)r_1$

$d = (z-1)(z-2)r_2+2r_2$

We will prove $\epsilon \le 0.5$ by showing that (1) $a/c \le 0.5$ and (2) $b/d \le 0.5$.

(1) $a/c = 2r_1/2(z-1)r_1$

$= 1/(z-1) \le 0.5$ since $z \ge 3$.

(2) $b/d = 2(z-2)r_2/((z-1)(z-2)r_2+2r_2)$

$= 2(z-2)/((z-1)(z-2)+2)$

We want to show $2(z-2)/((z-1)(z-2)+2) \le 0.5$. That is

$4(z-2) \le (z-1)(z-2)+2$

$z^2-7z+12 \ge 0$

Let $f(z) = z^2-7z+12 = (z-3)(z-4)$.

Since $f(z)$ is a convex function of z and since z is integer valued, then $f(z) = 0$ if $z = 3$ or 4 and $f(z) > 0$ otherwise. Hence, $b/d \leq 0.5$. Combining (1) and (2), we obtain $\epsilon \leq 0.5$ for $z = z \geq 3$.

In order to show that the bound is tight, consider the problem with the following data:

Job : j	1	2	3	4
Processing time : p_j	1	1	R_2	R_2

Schedule S obtained after applying Algorithm 2.2:

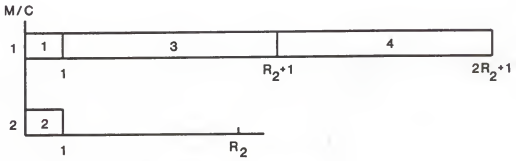


Figure 2-7. Schedule S.

$F(S) = 4 + 3R_2$

Optimal schedule S^*:

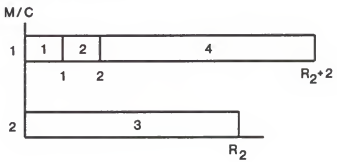


Figure 2-8. Schedule S^*.

$F(S^*) = 5 + 2R_2$

$\epsilon = ((2+3R_2)-(5+2R_2))/(5-2R_2) \approx 0.5$ as R_2 approaches infinity. Hence the error bound is tight. Q.E.D.

However, as the following empirical analysis shows, the average error obtained is never more than 0.1. We will next provide the empirical results of the heuristic compared to the optimal solutions as would have been obtained using the dynamic programming algorithm provided above (Algorithm 2.1).

Empirical Analysis

Each of the following problems (each row) is a replication of 100 randomly generated problems. The capacity tightness is specified as the ratio of R_2 to the sum of job processing times. The smaller this value, the sooner Machine 2 becomes unavailable (smaller R_2). Furthermore, for each problem, the job processing times are randomly generated to follow a uniform distribution over (MinP,MaxP). The deviation is calculated as the difference of the heuristic solution to the optimal solution over the optimal solution. The deviations over all 100 randomly generated problems are averaged to give the entries in the last column. Tables 2-1, 2-2, 2-3, and 2-4 illustrate the results for the 10-, 20-, 30-, and 40-job problems respectively.

The empirical study resulted in a maximum average deviation of the heuristic from the optimal solutions of less than 0.1. Averaging the entries of the fifth columns of the four tables, we obtain an overall average of 0.03.

Table 2-1. 10-Job CSFT Problems.

Number of Jobs	Capacity Tightness	MinP	MaxP	Average Dev.
10	0.10	5	10	0.024
10	0.10	5	20	0.056
10	0.10	5	30	0.063
10	0.10	5	40	0.075
10	0.20	5	10	0.034
10	0.20	5	20	0.080
10	0.20	5	30	0.081
10	0.20	5	40	0.097
10	0.30	5	10	0.021
10	0.30	5	20	0.049
10	0.30	5	30	0.060
10	0.30	5	40	0.072
10	0.40	5	10	0.013
10	0.40	5	20	0.008
10	0.40	5	30	0.012
10	0.40	5	40	0.012

Table 2-2. 20-Job CSFT Problems.

Number of Jobs	Capacity Tightness	MinP	MaxP	Average Dev.
20	0.10	5	10	0.035
20	0.10	5	20	0.028
20	0.10	5	30	0.044
20	0.10	5	40	0.033
20	0.20	5	10	0.057
20	0.20	5	20	0.026
20	0.20	5	30	0.048
20	0.20	5	40	0.046
20	0.30	5	10	0.039
20	0.30	5	20	0.035
20	0.30	5	30	0.029
20	0.30	5	40	0.025
20	0.40	5	10	0.009
20	0.40	5	20	0.022
20	0.40	5	30	0.030
20	0.40	5	40	0.032

Table 2-3. 30-Job CSFT Problems.

Number of Jobs	Capacity Tightness	MinP	MaxP	Average Dev.
30	0.10	5	10	0.029
30	0.10	5	20	0.026
30	0.10	5	30	0.022
30	0.10	5	40	0.023
30	0.20	5	10	0.011
30	0.20	5	20	0.024
30	0.20	5	30	0.030
30	0.20	5	40	0.035
30	0.30	5	10	0.017
30	0.30	5	20	0.030
30	0.30	5	30	0.029
30	0.30	5	40	0.024
30	0.40	5	10	0.014
30	0.40	5	20	0.026
30	0.40	5	30	0.019
30	0.40	5	40	0.013

Table 2-4. 40-Job CSFT Problems.

Number of Jobs	Capacity Tightness	MinP	MaxP	Average Dev.
40	0.10	5	10	0.008
40	0.10	5	20	0.019
40	0.10	5	30	0.019
40	0.10	5	40	0.018
40	0.20	5	10	0.021
40	0.20	5	20	0.021
40	0.20	5	30	0.018
40	0.20	5	40	0.021
40	0.30	5	10	0.011
40	0.30	5	20	0.013
40	0.30	5	30	0.022
40	0.30	5	40	0.022
40	0.40	5	10	0.016
40	0.40	5	20	0.004
40	0.40	5	30	0.008
40	0.40	5	40	0.010

It is also interesting to see if we can define the worst case error bound of the algorithm in terms of the tightness of the gap defined as the difference between the capacity of Machine 2 and the sum of processing times of jobs assigned to the machine by the heuristic. Let $\delta = R_2 - \Sigma_{i\in B_2} p_i$ and let δ/r_2 be the measure of tightness of the gap (note that $0 < \delta/r_2 < 1$ and if $\delta/r_2 = 0$, then from part (2) of Lemma 2.5, the sequence obtained is optimal). We will show that even when $|A_1|$ is small, if the gap is tight (i.e., δ/r_2 is small), the error bound is much less than 0.5.

Similar to the proof of part (2) of Lemma 2.5, we can show that

$$F(S)-F(S^*) \leq (z-1)\delta \tag{2.8}$$

Equivalently, the lower bound of the optimal value of the schedule can be rewritten as

$$\begin{aligned} F(S') &\geq ((z-1)(z-2)/2)r_2 + r_2 \\ &= (z^2-3z+4)r_2/2 \end{aligned} \tag{2.9}$$

Combining equations (2.8) and (2.9), we obtain

$$\begin{aligned} \epsilon &\leq F(S)-F(S^*)/F(S') \\ &\leq \left(\frac{2(z-1)}{z^2-3z+4}\right)\left(\frac{\delta}{r_2}\right) \end{aligned} \tag{2.10}$$

From equation (2.10), we can see the algorithm yields a schedule that is closer to optimal as δ/r_2 approaches zero (i.e., as the gap becomes tighter). The following table shows the upper bounds on the values of ϵ in terms of z and δ/r_2.

Table 2-5. Upper Bounds on ϵ for CSFT Problem Given z and δ/r_2.

z	δ/r_2			
	1/2	1/4	1/8	1/16
2	0.50	0.25	0.13	0.06
3	0.50	0.25	0.13	0.06
4	0.38	0.19	0.09	0.05
5	0.29	0.14	0.07	0.04

From the above table, we can see that even when $z = 2$, $\epsilon \leq 0.06$ if the tightness of the gap is at least 1/16 (i.e., $\delta/r_2 = 1/16$). Hence, for practical purpose, if either z is large or the gap is tight, we can expect the algorithm to perform well. Otherwise, we can either use some improvement procedures such as the one proposed by Kao and Elsayed (1989) in conjunction with Algorithm 2.2 or we can use the dynamic programming algorithm (Algorithm 2.1) to solve the problem optimally. In the next section, we will investigate the scheduling problem with preventive maintenance.

Preventive Maintenance Problem

The problem that we will now study involves the two-parallel identical machines preventive maintenance problem. We again assume that all machines are initially and simultaneously available. Note that $R_1 \leq \Sigma_j p_j$ for otherwise the problem becomes unconstrained on Machine 1. Similarly, we assume that $R_2 \leq \Sigma_j p_j$. Furthermore, after the maintenance on a machine has been completed, it becomes

continuously available. We only allow each machine to be taken out of production line once, at a pre-specified time, for preventive maintenance purpose.

This maintenance is done on a rotation basis. That is, we do not allow any two or more machines to be simultaneously maintained. Instead, we assume that as soon as one machine finishes, another machine will start its maintenance. The amount of time required to provide such maintenance is assumed to be the same for both machines. This is repeated until all machines in the production system have been maintained.

The justification behind such assumptions is the usual practice in industry of maintaining machines at predetermined times. The rotation assumption directly relates to the situation when there is only one maintenance crew available. Preventive maintenance usually requires a definite set of operations, which depends on the nature of the machine being maintained. Since the machines are assumed to be identical, the assumption of equal maintenance time for each machine is a realistic one. Again, without the preventive maintenance, the problem of minimizing the sum of job flow times can be optimally solved by the SPT algorithm.

Adiri et al. (1989) have looked at the one-machine problem. In this section, we will first provide a shorter NP-Completeness proof for the one-machine deterministic breakdown problem discussed in Adiri et al. (1989). We also show that the worst case error bound for the SPT heuristic is 2/7. Furthermore, we will provide an

example to show that the bound is tight, which also serves as a counter-example to the 1/4 error bound provided by Adiri et al. (1989).

We have also obtained some results for the two-parallel machine case. Specifically, we will provide a pseudo-polynomial dynamic programming algorithm to solve the two machines problem optimally. The performance of the SPT algorithm is studied next.

NP-Completeness of the One-Machine Maintenance Problem

We will now provide a simpler NP-Completeness proof for the one-machine case than the one provided by Adiri et al. (1989). Let R = R_1 and L be the length of maintenance, both of which are known in advance. This proof is similar to the one we propose for the capacitated machine problem. We will transform the even-odd partition problem (as described in the previous section), which is known to be NP-Complete (Garey et al., 1988), in polynomial time into the problem.

The corresponding one-machine problem can be constructed as follows:

Number of jobs:	$2n+1$
Processing times:	$p_i = M+x_i, \quad i=1,\ldots,2n$
	$P_{2n+1} = P$
Maintenance time:	$R = nM+Z$
Length of maintenance:	$L = M$
Sum of flow times:	$y = \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + 2n(n+1)M + nZ + (2(nM+Z)+M) + P$

where $Z = (\sum_{i=1}^{2n} x_i)/2$, $M > 2nZ$, and $P > \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + 2n(n+1)M + nZ + (2(nM+Z)+M)$. Note that $2P > y$.

Let sets B and A denote the sets of jobs completed before and after the breakdown respectively. Consider sets B and A' where A' is the set of jobs completed after the breakdown excluding the last job (see Figure 2-9).

Question: *Does there exist a nonpreemptive schedule S^* such that the flow time of the last job in set B is not greater than R and $F(S^*)$, the total flow times of all jobs in S^*, is no more than y?*

Remark: Note that in the above instance, $0 < p_1 < p_2 < \cdots < p_{2n+1}$.

Lemma 2.6: If there exists a solution to the even-odd partition problem, then there exists a schedule S^* for the breakdown problem with total flow times $F(S^*) = y$.

Proof: Let sets X_1 and X_2 be the solution to the even-odd partition problem. We can construct a schedule S^* of the jobs as shown in Figure 2-9, where if x_i is in set X_1 (set X_2), then job i is in set B (resp. set A') and all the jobs in B and A' are in SPT.

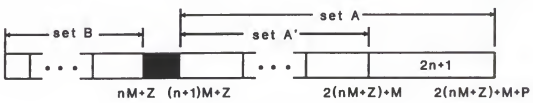


Figure 2-9. Schedule S^*.

Since the solution to the even-odd partition exists, we have

$$\Sigma_{i\in X_1} x_i = \Sigma_{i\in X_2} x_i = Z$$

This implies there exists sets B and A' consisting of the first 2n jobs such that

$$\Sigma_{i\in B} p_i = nM + \Sigma_{i\in X_1} x_i = nM+Z$$

$$\Sigma_{i\in A'} p_i = nM + \Sigma_{i\in X_2} x_i = nM+Z$$

Let $p_{(i)}$ be the processing time of the i^{th} job in set A'. The flow time of job (i) is given by $[(n+1)M+Z + \Sigma_{j\le i} p_{(j)}]$. The sum of flow times of jobs in set A' is therefore given by

$$\Sigma_{j\in A'} C_j(S^*) = n[(n+1)M+Z] + \sum_{i=1}^{n}(n-i+1)p_{(i)}$$

Hence, we obtain

$$F(S^*) = \Sigma_{j\in B} C_j(S^*) + \Sigma_{j\in A'} C_j(S^*) + C_{2n+1}(S^*)$$

$$= \sum_{i=1}^{n}(n-i+1)(p_{2i-1}+p_{2i}) + n[(n+1)M+Z] + [(2n+1)M+2Z+p_{2n+1}]$$

$$= \sum_{i=1}^{n}(n-i+1)[(M+x_{2i-1})+(M+x_{2i})]+n[(n+1)M+Z]+[2(nM+Z)+M]+P$$

$$= \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + 2n(n+1)M + nZ + [2(nM+Z)+M] + P$$

$$= y \qquad \text{Q.E.D.}$$

Lemma 2.7: Let S^* be a feasible schedule for the problem with $F(S^*) \le y$, then in S^*, the following must be true:

(1) job 2n+1 is processed last among all jobs,

(2) $|B| = |A'| = n$,

(3) exactly one of jobs 2i-1 or 2i (i = 1,...,n) is in B and the other is in A' and $\Sigma_{j\in B}\ p_j = \Sigma_{j\in A'}\ p_j = nM+Z$ (or equivalently, $\Sigma_{j\in B}\ x_j = \Sigma_{j\in A'}\ x_j = Z$).

Proof: Part (1): It can be easily checked that $p_{2n+1} > R$. Hence job 2n+1 must be processed after breakdown. Now, suppose job 2n+1 is not processed last. This implies that there is another job, say job j, that is processed after job 2n+1. Then

$$\begin{aligned} F(S^*) &\geq p_{2n+1} + (p_{2n+1} + p_j) \\ &= 2p_{2n+1} + p_j \\ &= 2P + p_j \\ &> y \end{aligned}$$

Hence, for $F(S^*) \leq y$, job 2n+1 has to be processed last (after set A'). From now on we will only consider sets B and A', which contain the first 2n jobs.

Part (2): By definition, $R = nM+Z$ and $p_i > M$ for all job i. Hence set B cannot contain more than n jobs. Consider a feasible schedule S' whereby $|B| = n' \leq n-1$ and $|A'| = 2n - n' = n'' \geq n+1$. Let $p_{[i]}$ be the processing time of the i^{th} job in schedule S'. Then

$$\begin{aligned} F(S') &= \sum_{i=1}^{n'} (n'-i+1)\ p_{[i]} + \sum_{i=1}^{n''} (n''-i+1)\ p_{[n'+i]} + \\ &\quad (n'')[(n+1)M+Z) + C_{2n+1}(S') \\ &> \frac{n'(n'+1)}{2} M + \frac{n''(n''+1)}{2} M + n''[(n+1)M+Z) \\ &\quad + (n+1)M + Z + n''M + P \end{aligned}$$

$$\geq n(n+1)M + M + (n+1)((n+1)M+Z) + (n+1)M + Z + (n+1)M + P$$

$$= n(n+1)M + M + n((n+1)M+Z) + ((n+1)M+Z) + 2(n+1)M + Z + P$$

$$= 2n(n+1)M + nZ + 2(nM+Z) + (n+4)M + P$$

$$= 2n(n+1)M + nZ + (2(nM+Z)+M) + P + (n+3)M$$

$$> y \quad \text{since } \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) \leq 2nZ < M < (n+3)M.$$

Therefore, in order to have $F(S) \leq y$, we must have $|B| = |A'| = n$.

Part (3): By part (2), we know that in schedule S^*, both sets B and A' contain n jobs.

$$F(S^*) = \Sigma_{i\in B}\, C_i(S^*) + \Sigma_{i\in A'}\, C_i(S^*) + C_{2n+1}(S^*)$$

$$= np_{[1]}+(n-1)p_{[2]}+\cdots+p_{[n]} + n(nM+Z+M)$$
$$+np_{[n+1]}+(n-1)p_{[n+2]}+\cdots+p_{[2n]}$$
$$+((nM+Z)+M+(p_{[n+1]}+p_{[n+2]}+\cdots+p_{[2n]})+P)$$

$$= n(M+x_{[1]})+(n-1)(M+x_{[2]})+\cdots+(M+x_{[n]}) + n((n+1)M+Z)$$
$$+n(M+x_{[n+1]})+(n-1)(M+x_{[n+2]})+\cdots+(M+x_{[2n]})$$
$$+((nM+Z)+M+(nM+x_{[n+1]}+x_{[n+2]}+\cdots+x_{[2n]})+P)$$

$$= n(x_{[1]})+(n-1)(x_{[2]})+\cdots+x_{[n]}$$
$$+n(x_{[n+1]})+(n-1)(x_{[n+2]})+\cdots+x_{[2n]}$$
$$+(x_{[n+1]}+x_{[n+2]}+\cdots+x_{[2n]})$$
$$+n((n+1)M+Z)+n(n+1)M+(2nM+Z+M+P)$$

$$= n(x_{[1]})+(n-1)(x_{[2]})+\cdots+x_{[n]}$$
$$+n(x_{[n+1]})+(n-1)(x_{[n+2]})+\cdots+x_{[2n]}$$
$$+x_{[n+1]}+x_{[n+2]}+\cdots+x_{[2n]} + (2n^2+4n+1)M+(n+1)Z+P$$

For $F(S^*) \le y = \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + 2n(n+1)M + nZ + (2(nM+Z)+M)+P$, we must have

$$n(x_{[1]})+(n-1)(x_{[2]})+\cdots+x_{[n]} \tag{2.11}$$

$$+n(x_{[n+1]})+(n-1)(x_{[n+2]})+\cdots+x_{[2n]} \tag{2.12}$$

$$+x_{[n+1]}+x_{[n+2]}+\cdots+x_{[2n]} \tag{2.13}$$

$$\le \sum_{i=1}^{n}(n-i+1)(x_{2i-1}+x_{2i}) + Z$$

Since the coefficients of $x_{[i]}$'s in (2.11) and (2.12) are both decreasing, if we assign the two smallest x_i's (= x_1 or x_2) to either the first or the $(n+1)^{th}$ positions, the next two smallest x_i's (= x_3 or x_4) to either the second of the $(n+2)^{th}$ positions, etc., the following minimum cost is obtained

$$F_{min} = \sum_{i=1}^{n}(n-i+1)\ (x_{2i-1}+x_{2i})$$

$$\le n(x_{[1]})+(n-1)(x_{[2]})+\cdots+x_{[n]}$$
$$+n(x_{[n+1]})+(n-1)(x_{[n+2]})+\cdots+x_{[2n]}$$

Therefore, we must have $x_{[n+1]}+\cdots+x_{[2n]} \le Z$. However, a feasible schedule implies that $x_{[1]}+\cdots+x_{[n]} \le Z$. Hence, for equation (2.13), $x_{[n+1]}+\cdots+x_{[2n]} = Z$ since $x_{[1]}+\cdots+x_{[2n]} = 2Z$. This implies

$$\sum_{i=1}^{n}(n-i+1)\ (x_{2i-1}+x_{2i}) = n(x_{[1]})+(n-1)(x_{[2]})+\cdots+x_{[n]}$$
$$+n(x_{[n+1]})+(n-1)(x_{[n+2]})+\cdots+x_{[2n]}$$

Namely, the two smallest x_i's (= x_1 or x_2) have to be assigned to either the first or the $(n+1)^{th}$ positions, the next two smallest x_i's to either the second of the $(n+2)^{th}$ positions, etc.. Q.E.D.

Remark: Part (3) in Lemma 2.7 implies that there exists an even-odd partition. Combining Lemma 2.7 with Lemma 2.6 we have shown the following theorem.

Theorem 2.4: The one-machine preventive maintenance problem is NP-Complete.

Heuristic

We will now prove that the SPT heuristic has a tight worst case error bound of 2/7.

Theorem 2.5: The SPT heuristic has a tight worst case error of 2/7.

Proof: Consider a schedule S generated by the SPT algorithm and an optimal schedule S^* where $|X| = |B|$ and $|Y| = |A|$. Let δ and δ^* be the differences between R and the completion times of the last jobs processed before R in schedule S and S^* respectively (see Figures 2-10 and 2-11).

Figure 2-10. Schedule S.

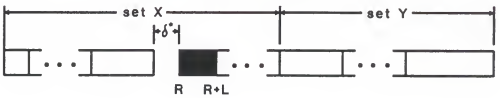


Figure 2-11. Schedule S^*.

Note that if $|A| = |Y| = 0$, then the problem becomes unconstrained and SPT is optimal. Furthermore, if $|B| = |X| = 0$, then $p_1 < R$. In this case, SPT is again optimal since all jobs have to be processed after the breakdown. Hence, we will only discuss the case with $|A| = |Y| \geq 1$ and $|B| = |X| \geq 1$.

We will first discuss the case $(\delta - \delta^*) \geq 0$. Note that $\Sigma_{j \in X} p_j \geq \Sigma_{j \in B} p_j$ and all jobs in set Y are completed after the breakdown since set B contains the smallest $|B|$ jobs. Let C_i and C_i^* be the completion times of the first job in set A and Y respectively and $F_V(\sigma)$ be the sum of flow times of set V in schedule σ. We can easily see that $C_i \leq C_i^* + (\delta - \delta^*)$. Hence

$$F_A(S) \leq F_Y(S^*) + |Y|(\delta - \delta^*)$$

However, $F_B(S) \leq F_X(S^*) - (\delta - \delta^*)$, since $\Sigma_{j \in X} p_j \geq \Sigma_{j \in B} p_j + (\delta - \delta^*)$ and set B is in SPT order. We obtain

$$\begin{aligned} F(S) &= F_B(S) + F_A(S) \\ &\leq F_X(S^*) + F_Y(S^*) + (|Y|-1)(\delta - \delta^*) \\ &= F(S^*) + (|Y|-1)(\delta - \delta^*) \end{aligned} \tag{2.14}$$

Note that all $p_j \geq \delta \geq (\delta - \delta^*)$ for all $j \in A$ and all jobs in A are in SPT. Hence

$$\begin{aligned} F(S^*) &\geq \left\{\frac{|A|(|A|+1)}{2}\right\} (\delta - \delta^*) \\ &= \left\{\frac{|Y|(|Y|+1)}{2}\right\} (\delta - \delta^*) \end{aligned}$$

Considering jobs in set B, if B = X, then the schedule S is optimal. We will, therefore consider the case B ≠ X. In this case, there exists a job j ∈ B, which in the optimal schedule S^*, is processed after the breakdown. Hence, the difference of the completion times of the last jobs in set B and set X is at least $(\delta-\delta^*)$. We therefore obtain the following lower bound on the total flow times of the optimal schedule S^*:

$$F(S^*) \geq \left\{\frac{|Y|(|Y|+1)}{2} + 1\right\} (\delta-\delta^*) \tag{2.15}$$

Combining equations (2.14) and (2.15), we obtain

$$\epsilon = \frac{F(S)-F(S^*)}{F(S^*)} \leq \frac{2(|Y|-1)}{|Y|(|Y|+1) + 2}$$

If $|Y| = 1$, $\epsilon = 0$ and S is optimal. For $|Y| = 2$, 3, 4, and 5, $\epsilon =$ 1/4, 2/7, 3/11, and 1/4 respectively. It can be checked that $\epsilon < 1/4$ for $|Y| > 5$. Therefore, the SPT algorithm has a worst case error bound of 2/7, which is greater than 1/4.

We will now show that in an optimal schedule, it is impossible to have $(\delta-\delta^*) < 0$. Suppose that $(\delta-\delta^*) < 0$, it can be seen from above discussion that $F_A(S) < F_Y(S^*)$. Also $F_B(S) \leq F_X(S^*)$ since set B contains the smallest $|B|$ jobs arranged in SPT. This implies that $F(S) = F_B(S) + F_A(S) < F_B(S) + F_Y(S^*) \leq F(S^*)$ since always $F_B(S) \leq F_X(S^*)$, contradicting to the optimality of S^*.

The following example shows that that the error is tight:

Job : j	1	2	3	4
Processing time : p_j	1	M	M	M

R = M and L = 1, where M is a very large number.

Schedule S obtained using the SPT heuristic is:

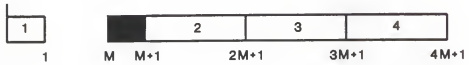


Figure 2-12. Schedule S.

F(S) = 9M+4

The optimal schedule S^* is:

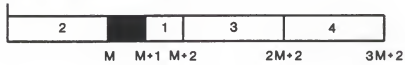


Figure 2-13. Schedule S^*.

$F(S^*) = 7M+6$

$\epsilon = [(9M+4)-(7M+6)]/(7M+6) \approx 2/7$ as M approaches infinity. Hence the error bound is tight. Q.E.D.

Dynamic Programming Algorithm for Two-Machine Problem

We will now provide a pseudo-polynomial dynamic programming algorithm to solve the two-machine problem, which together with the fact that it is NP-Complete, implies that the problem is NP-Complete in the ordinary sense.

Algorithm 2.3

Let $f(j,t_1,t_2)$ = minimum sum of job flow times if we have scheduled j jobs and the total processing times of jobs assigned to Machines 1 and 2 before R_1 and R_2 are t_1 and t_2 respectively,

$g_1(j,t_1,t_2)$ = completion time of the last job processed after the maintenance in Machine 1 in the solution of $f(j,t_1,t_2)$,

$g_2(j,t_1,t_2)$ = completion time of the last job processed after the maintenance in Machine 2 in the solution of $f(j,t_1,t_2)$.

Boundary condition:

$f(0,0,0) = 0$

$f(j,t_1,t_2) = \infty$ for $j = 0,\ldots,n$ and either $t_1 < 0$ or $t_2 < 0$

$g_1(j,t_1,t_2) = R_1+L$ for $j = 0,\ldots,n$, $t_1 \le 0$ and $t_2 \le 0$

$g_2(j,t_1,t_2) = R_2+L$ for $j = 0,\ldots,n$, $t_1 \le 0$ and $t_2 \le 0$

Recursive Relation:

$$f(j,t_1,t_2) = \min \begin{cases} f(j-1,t_1-p_j,t_2) + t_1 & (2.16) \\ f(j-1,t_1,t_2-p_j) + t_2 & (2.17) \\ f(j-1,t_1,t_2) + g_1(j-1,t_1,t_2) + p_j & (2.18) \\ f(j-1,t_1,t_2) + g_2(j-1,t_1,t_2) + p_j & (2.19) \end{cases}$$

$$g_1(j,t_1,t_2) = \begin{cases} g_1(j-1,t_1,t_2)+p_j & \text{if } f(j,t_1,t_2)\text{=equation (2.18)} \\ g_1(j-1,t_1,t_2) & \text{otherwise} \end{cases}$$

$$g_2(j,t_1,t_2) = \begin{cases} g_2(j-1,t_1,t_2)+p_j & \text{if } f(j,t_1,t_2)\text{=equation (2.19)} \\ g_2(j-1,t_1,t_2) & \text{otherwise} \end{cases}$$

where $j = 1,\ldots,n$

$t_1 = 0,\ldots,R_1$

$t_2 = 0,\ldots,R_2$

Solution:

$$F(S^*) = \min\{f(n,t_1,t_2): t_1 = 0,\ldots,R_1 \text{ and } t_2 = 0,\ldots,R_2\} \quad (2.20)$$

Complexity:

Since $j = 0,\ldots,n$, $t_1 = 0,\ldots,R_1$, and $t_2 = 0,\ldots,R_2$, the complexity of Algorithm 2.3 is $O(nR_1R_2) = O(n(\Sigma p_j)^2)$.

Justification of Algorithm 2.3

Let job j be the job currently considered. Job j can either be assigned to sets B_1 or B_2, or it can be assigned to neither one of the sets. If job j is assigned to set B_1, then the sum of flow times of the partial schedule up to job j will be the sum of flow times prior to assigning job j ($= f(j-1,t_1-p_j,t_2)$) and the flow time of job j ($= t_1$) resulting in equation (2.16).

On the other hand, if job j is assigned to set B_2, then the sum of flow times of the partial schedule up to job j will be the sum of flow times prior to assigning job j ($= f(j-1,t_1,t_2-p_j)$) and the flow time of job j ($= t_2$) resulting in equation (2.17).

The last scenario to consider is when job j is neither assigned to sets B_1 nor B_2. In this case job j is either assigned to set A_1, which implies equation (2.18) or assigned to A_2, which implies equation (2.19). If job j is assigned to A_1, then $g_1(j,t_1,t_2)$ will be updated by adding p_j to $g_1(j-1,t_1,t_2)$. Otherwise, $g_1(j,t_1,t_2)$ will remain the same as $g_1(j-1,t_1,t_2)$. Similarly for $g_2(j,t_1,t_2)$.

Chapter Summary

In this chapter, we have studied the problem of machines with restricted availability to minimize sum of job flow times. In particular, we present some results on three special cases of the general problem as stated in the introduction. For the rolling horizon problem, we prove that the SPT algorithm still yields optimal schedules. For the machine with prior commitment problem, we prove the problem to be NP-Complete, provide an $O(nR_2)$ dynamic programming algorithm, show that the SPT heuristic has a tight worst case error bound of 0.5, and show that on average, the average error is less than 0.1. For the one-machine preventive maintenance problem, we show that the problem is NP-Complete, show that the SPT heuristic has 2/7 worst case error bound, and propose a dynamic programming algorithm of complexity $O(n(\Sigma p_j)^2)$ for the two parallel machines problem.

CHAPTER 3
SCHEDULING WITH NONREGULAR MEASURE OF PERFORMANCE

Introduction and Motivations

In this chapter, we will study the n-job nonpreemptive static production scheduling problems to minimize a nonregular measure of performance, namely the sum of job earliness and tardiness penalties. The earliness (tardiness) of a job is defined as the length of time the job is completed before (after) its due date. There are basically two types of due dates: (1) exogenous or externally specified (by customers) due dates and (2) endogenous or internally specified (by company) due dates. As we have mentioned, in some industrial settings, due dates can sometimes be negotiable and hence can be treated as decision variables within the boundary of the scheduling problem.

This performance measure, the sum of job earliness and tardiness, is a direct consequence of the JIT concept in which we discourage jobs that are completed either too early or too late by penalizing them. With the advent of the JIT manufacturing practices, customers may refuse to take deliveries of products that are completed early. Thus, finishing the jobs early would incur inventory holding costs in the form of tied-up capital and warehouse space. On the other hand, if the jobs are completed late, penalties which include express deliveries and loss of customer goodwill will

be imposed. These penalties usually depend on the degree of tardiness of the jobs. Hence, we assign earliness and tardiness weights to the job depending on its value and importance. The total penalty increases proportionally with the job earliness or tardiness.

In this chapter we will present the results for the exogenous common due date earliness-tardiness problems with unit weights (weights = 1 for all jobs). This is sometimes referred to as the sum of absolute deviations of completion times about a common due date. We will propose a new heuristic for this problem and we will prove that it has a tight worst case error bound of 0.5. So far in the literature, the heuristics that exist for the problem can be shown to have arbitrarily bad worst case performances. We will also provide some empirical results that show the average performance of the algorithm. We will now give some definitions and terminologies.

Definitions and Overview of Terminologies

Most of the literature that exists on the earliness-tardiness problem assumes the existence of a common due date. Unless specifically mentioned otherwise, the discussions are for the common due date problems. As we have mentioned, there are two basic types of due dates, exogenous and endogenous. In the nonregular performance literature, the first type of due date is also commonly known as the case of restricted (or restrictively large) due date while the second is also called the case of unrestricted due date. We shall use these two terminologies interchangeably.

The earliness and tardiness of a job are defined as the length of times the job is completed before and after its due date, respectively. Depending on the importance of the individual job, an earliness and a tardiness weights will be assigned to each job. For the endogenous due date problem, the objective will be to find the due date and the corresponding schedule such that the sum of the weighted penalties is minimized. For the exogenous due date problem, the objective is only to find the schedule which minimizes the total penalties. To achieve this end, we will define the following additional notation:

D = common due date

α_j = earliness penalty of job j per unit time early

β_j = tardiness penalty of job j per unit time late

The objective of the problems is to find the schedule (and the due date, for the endogenous problem), that minimizes the following total costs:

$$\sum_{j=1}^{n} \alpha_j E_j + \beta_j T_j \tag{3.1}$$

Note that when $\alpha_j = 0$ and $\beta_j = 1$, the problem becomes the well known sum of tardiness problem which has been proven to be NP-Complete in the ordinary sense by Du and Leung (1990). Lawler (1977) provides a pseudo-polynomial dynamic programming algorithm to solve the problem. For the case $D_j = D$ (common due date) and $\alpha_j = 0$, Lawler and Moore (1969) propose a pseudo-polynomial dynamic programming algorithm.

From now on we will assume that jobs are numbered in nondecreasing order of processing times (i.e., $p_1 \leq p_2 \leq \cdots \leq p_n$). We will now review some of the literature that exists on the nonregular measure of performance. Even though we will only present the results for the exogenous common due date problem with unit weights, we will also review the literature for the other cases of due dates and weights.

Literature Review

The classical due date related measures of performance are maximum lateness, mean lateness, maximum tardiness, mean tardiness, and number of tardy jobs. Sen and Gupta (1984) provide an excellent survey of these classical due date related measures.

One of the first studies on due date related measures was conducted by Jackson (1955). In that paper, he proves that when all jobs are initially available, the solution to the exogenous problem of minimizing the maximum lateness, defined as $\max_j (D_j - C_j)$, on a single machine is given by scheduling the jobs in nondecreasing order of their due dates; i.e., the Earliest Due Date (EDD) rule. Since then, many researchers have studied many variations of this problem under various assumptions and conditions.

The earliest discussion on the earliness-tardiness problem can be found in Sidney (1977). He considers a one-machine scheduling problem in which penalties are incurred when jobs are started before their target start times (early) or completed after their due dates

(tardy). The objective is to minimize the maximum of these penalties.

For excellent reviews of existing literature on the earliness and tardiness problems, the readers are referred to survey articles by Raghavachari (1988) and Baker and Scudder (1990). For problems in which due dates are decision variables, interested readers are referred to a noteworthy survey by Cheng and Gupta (1989). In the following review, we will classify the body of literature that exists on the earliness-tardiness problems as either common due date or different due dates problems.

Common Due Date ($D_j = D$)

Most of the existing literature on earliness-tardiness penalties looks at static common due date problems. Bagchi (1989) gives the following four justifications for the validity of the common due date assumption. The first situation is when the jobs are sub-assemblies required for a final product. The second scenario is in the context aggregate shipping to lower shipping costs. The third justification is in the kitting problem when the jobs are part of a kit. The final scenario concerns a customer order that consists of multiple units of the same product.

Within this category, we further classify the literature in terms of the job earliness and tardiness weights. The four weight classifications are as follows:

(1). Unit Weight Case ($\alpha_j = \beta_j = 1 \ \forall \ j$)

(2). Constant Weight Case ($\alpha_j = \alpha$ and $\beta_j = \beta \ \forall \ j$)

(3). Symmetric Weight Case ($\alpha_j = \beta_j \ \forall \ j$)

(4). General Weight Case ($\alpha_j \neq \beta_j \ \forall \ j$)

The simplest common due date problem with unit weights has been proven to be NP-Complete by Hall et al. (1989) for the unrestricted due date case. Hence, some researchers restrict D to be large enough (say, $D \geq \Sigma_j \ p_j$) so as not to constrain the schedule. The problem with restricted D can in fact be viewed as an endogenous due date problem since if D is large enough (so as not to constrain the schedule), it can always be reduced to the minimum value possible while still maintaining the nonconstraining property. This is equivalent to the endogenous due date problem. The difficulty arises when we have to decide whether the due date is tight or loose.

The endogenous (restrictively large) due date problems can still be solved in polynomial time for the constant weight case. Other cases require the use of pseudo-polynomial dynamic programming algorithms. For the exogenous (unrestricted) due date, no polynomial time algorithm exists. Hence, either enumerative procedures such as dynamic programming and branch and bound or heuristic algorithms are used. We will now discuss the relevant publications on these problems.

Unit Weight Case

Kanet (1981) is the first to consider the one-machine problem of minimizing the sum of absolute deviations of completion times about a restrictively large common due date. He provides an optimal

algorithm and also shows the similarities between this problem and the two-machine sum of job flow times problem.

Kanet's results are extended by Sundararaghavan and Ahmed (1984) to the case of m-parallel identical machines with the restricted due date assumption. An optimal algorithm, capable of producing multiple optimal schedules, that consists of a partitioning and a scheduling procedure is provided. They also give a heuristic procedure for the one-machine case of small D with the restriction that the first job starts at time zero (zero start time schedule). They provide some empirical results but not a theoretical worst case error bound.

Bagchi et al. (1986) consider the same restricted problem and provide an optimal algorithm for determining multiple optimal schedules. They also provide an implicit enumeration procedure that uses a number optimal properties of for the case of unrestricted due date.

Similar to Bagchi et al. (1986), Hall (1986) extends Kanet's (1981) algorithm by providing an algorithm that will find alternate optimal solutions. It also extends the result by providing an algorithm for the case of multiple parallel machines similar to the one provided by Sundararaghavan and Ahmed (1984). Hall explains the algorithm in terms of job positional weights. Again, he assumes the due date to be restrictively large.

One of the properties that proves to be useful time and time again is the optimal V-shaped property. In a common due date optimal

schedule, the set of jobs that are processed before the job with the smallest processing time are arranged in LPT manner (nonincreasing order of processing times) while jobs that follow the job with the smallest processing time are arranged in SPT (nondecreasing order of processing times). In a short note, Raghavachari (1986) proves that the V-shape optimality property holds true for any common due date absolute deviation problems, a result that is shown by Eilon and Chowdhury (1977) for the variance of completion times problem. Lee et al. (1991) have further shown that the V-shape property still holds even for the case of arbitrary weights as long as the ratios of the weights to the processing times follow an agreeable condition.

As was mentioned, the restriction that $D \geq \Sigma_j p_j$ may be more restrictive than necessary. Cheng (1987a) proposes a tighter common due date that still produces the same optimal sequence.

Emmons (1987) investigates the absolute deviation and the constant weight problems on parallel identical and uniform machines. He also looks at the secondary criteria of minimizing the makespan. Cheng (1988a) studies the optimal common due date problem to minimize the sum of due date and earliness-tardiness costs using a duality approach.

Bector et al. (1988) solves the endogenous absolute deviation problem using a linear goal programming approach starting from an arbitrary sequence. They obtain the optimal common due date and its corresponding sequence.

Szwarc (1989) studies the problem in the context of whether the start time is arbitrary or fixed. This is equivalent to the problem with large or small due dates. He proposes a branch and bound algorithm for the case of fixed start time.

Hall et al. (1989) present one of the first NP-Complete proof for the earliness-tardiness problem. They prove that the restricted common due date case is NP-Complete and provide a pseudo-polynomial dynamic programming algorithm of complexity O(nD) to solve the problem.

Kubiak et al. (1990) show the equivalence of the sum of flow times and the sum of absolute deviations problems for the uniform machines with restricted common due date. They also show that for unrelated machines, the problem can be reduced to a transportation problem.

Constant Weight Case

Panwalkar et al. (1982) consider the endogenous due date problem with the added due date penalty. Hence, the optimal due date cannot be too large since there is a nonzero penalty associated with assigning a large due date. A general discussion on different due dates for each job is also given. Cheng (1986) subsequently solves the problem using the linear programming duality approach.

Bagchi et al. (1987a) examine the one-machine problem where the objectives are to minimize the weighted sum of absolute deviations (WSAD) and the weighted sum of squared deviations (WSSD). Different penalty rates are assessed to jobs that are tardy or early. The

authors consider two versions of each problem; the unconstrained version where an increase in due date does not yield a further decrease in the objective function value (large due date) and the constrained version where the objective function is decreased by increasing the due date (small due date). They provide a constructive algorithm for the unconstrained WSAD problem and implicit enumeration procedures based on some dominance properties for the other three problems.

Baker and Chadowitz (1989) extends the one-machine tight due date heuristic proposed by Sundararaghavan and Ahmed to the constant weight case. They also relax the assumption of zero start time and provide a neighborhood search procedure. Again, they only present some empirical results.

Krieger and Raghavachari (1989) consider the problem of scheduling n jobs on m identical parallel machines so as to minimize a penalty (loss) function. Deterministic and random processing times are considered. For the deterministic and linear loss function case, the authors distinguish the two cases of due date: endogenous and exogenous. For the former case, the authors provide an optimal algorithm and for the latter case, a heuristic without any performance evaluation is provided. The general V-shaped property of the general loss function is established. The quadratic loss function is considered for the one-machine case. For the random processing times, they provide a local optimal condition to be used

in a heuristic for the case of one-machine, linear loss, and a common due date.

Raghavachari (1989) proposes a branch and bound algorithm for the exogenous due date problem. The lower bound is obtained by solving the endogenous problem. He also suggests, for cases in which the number of jobs is large, terminating the branch and bound procedure after a predetermined value for the difference between the lower and upper bounds is attained.

Hoogeveen et al. (1990) propose a new lower and upper bounds for the branch and bound algorithm (for example, the one proposed by Bagchi et al., 1987a) to solve this particular case. These bounds can be calculated in polynomial time. If these bounds do not concur, they show that these bounds can be refined by solving a subset-sum problem using a pseudo-polynomial algorithm.

Lee et al. (1991) look at the constant weight problem with an additional penalty for weighted number of tardy jobs. For the endogenous due date case, they provide a polynomial algorithm. For the exogenous due date case, they propose a dynamic programming algorithm of complexity $O(n(D+p_{max}))$.

Symmetric Weight Case

Cheng (1985) investigates the common due date determination for the one-machine problem of minimizing the weighted sum of absolute latenesses. He uses the dual problem of the LP formulation to obtain the optimal due date.

Cheng (1987c) looks at the determination of the common due date and the optimal sequence using a partial enumeration procedure of complexity $O(2^n n^2)$. He further reduces the complexity to $O(n^{0.5}2^n)$ in Cheng (1990b).

Hall and Posner (1989) consider the problem on a single processor around a sufficiently large common due date. They provide some optimal properties, the first NP-Completeness proof for the common due date earliness-tardiness problems, and the first pseudo-polynomial dynamic programming algorithm of complexity $O(n\Sigma_j\ p_j)$ to solve the problem.

Hoogeveen and van de Velde (1989) also propose a pseudo-polynomial dynamic programming algorithm for the case of small due date of complexity $O(n^2D)$. They provide a simpler NP-Completeness proof than the one given by Hall et al. (1989).

Cheng and Kahlbacher (1989) look at the problem of minimizing the constant weighted earliness-tardiness penalties with the addition of a linear due date penalty (similar to Panwalkar et al. (1982)). They propose a pseudo-polynomial dynamic programming algorithm of complexity $O(n\Sigma_j\ p_j)$ based on some optimality properties.

General Weight Case

Quaddus (1987) attaches the due date penalty to the earliness-tardiness penalties similar to the problem studied by Panwalkar et al. (1982). He makes use of duality theory to solve the problem. However, he only deals with the common due date determination and neglects the sequencing aspect of the problem.

Rachamadugu (1990b) studies the case where the earliness and tardiness penalties are proportional to the processing times, possibly with different constants of proportionality. He studies the different due date case and use the characteristics of the dominant solution to characterize optimal solutions to the common due date problem. He also proves that the LPT algorithm yields an optimal solution for the problem he considers.

Lee et al. (1991) look at the general weight problem. The generality of the weights is only restricted by a realistic assumption on the ratios of the job processing times to the earliness-tardiness weights. In particular, they assume that $p_i/\alpha_i \geq p_j/\alpha_j$ also implies $p_i/\beta_i \geq p_j/\beta_j$. A practical implication of this condition is that if job i is relatively more important than job j, both its earliness and tardiness weights will be greater than that of job j. They call such assumption the agreeable-ratio condition. In addition to the "agreeable" weighted tardiness and earliness penalties, they also incorporate the weighted penalties due to the number of tardy jobs in the total penalty function. For the exogenous problem, they propose a dynamic programming algorithm of complexity $O(n\Sigma_j\ p_j)$. For the endogenous problem, they provide a more efficient dynamic programming algorithm of complexity $O(n^2D)$.

Davis and Kanet (1989) investigate the more general nonregular convex completion costs. They find that it is not much more difficult to design an enumerative search for problems with a convex, nonregular measure than it is for those with regular measures.

Furthermore, they also propose an efficient timetabling procedure which can be embedded in an enumerative algorithm allowing the search to be conducted over the domain of job permutations.

Kahlbacher (1989) proposes a pseudo-polynomial dynamic programming algorithm of complexity $O(n\Sigma_j p_j)$ for the general penalty function with unit weights and exogenous common due date. The penalty function is only required to achieve its minimum if the job meets its due date and to have the property that it decreases with increasing job earliness and increases with increasing job tardiness. He also provides a fully polynomial approximation scheme for a subclass of penalty functions.

Different Due Dates

Seidman et al. (1981) investigate the method of assigning different due dates to the jobs to minimize the penalties associated with the lead times, earliness, and tardiness of the jobs. The job lead time is defined as the length of time the assigned due date exceeds what the customer expected. The earliness and tardiness penalties are constants. They prove that the SPT algorithm when used in conjunction with their due date assignment procedure provides an optimal solution.

Fry et al. (1987) propose heuristics based on pairwise interchange procedures for the problem of different due dates with arbitrary weights. They also provide an optimal one-pass procedure to optimally insert idle times since when the due dates are specified

as part of the problem inputs, the optimal solution may contain idle times.

Garey et al. (1988) consider the one-processor scheduling problem where each job has a specified processing time and a preferred starting time (or equivalently a preferred finishing time). The objective is to find a schedule that minimizes the sum of the absolute discrepancies from the preferred starting times or the maximum of such discrepancies. This is equivalent to the earliness-tardiness problem with different due dates. They show the NP-completeness of the first objective and provide an efficient algorithm that finds minimum cost schedules whenever the tasks either all have the same length or are required to be processed in a given fixed sequence. For the second objective, they provide an efficient algorithm that finds minimum cost schedules in general.

Ow and Morton (1989) use the filtered beam search procedure developed in Ow and Morton (1988) to solve the same problem as Fry et al. (1987). They derive some adjacency conditions which are used in the procedure. However, they assume that jobs are processed consecutively without idle times. Such assumption is only relevant when the due dates are "tight." Moreover, the heuristic approach is difficult to study analytically.

Rachamadugu (1990a) studies the problem with different due dates and derives an optimality condition and uses it to find dominant solutions. He also allows idle times to be inserted in the schedule.

He also shows that using the SLK and the TWK due date assignment methods, the SPT algorithm yields an optimal solution.

Cheng (1990a) provides a dynamic programming algorithm to minimize sum of earliness-tardiness penalties given that the jobs have different due dates. However, this algorithm becomes impractical as the number of jobs becomes large.

In a somewhat related problem, Bagchi (1989) studies the problem when the jobs are partitioned into a number of multi-job customer orders with all jobs in an order having a common due date. The jobs that belong to specific customer orders are known in advance. The penalty function consists of the sum of linear earliness and tardiness penalties and linear penalties associated with the lead times of the customer orders. The lead time of a customer order is defined as the completion time of the last job in that order. The objective is to determine the due dates and schedules to minimize the penalty function.

Chand and Chhajed (1989) investigate similar problems as Bagchi (1989). Each customer order is assigned an order due date. However, instead of associating penalties with lead times, they use order due dates. They study two different cases. In the first case, they assume the number of jobs in a customer order is known but the specific jobs are not. In the second case, both the number and the specific jobs assigned to the orders are not known. The objective is to find the order due dates and the associated sequence so as to minimize the total penalties.

Other Variations

Cheng (1988b) solves the static one-machine problem of minimizing earliness and tardiness penalties about a common due date. However, he assumes that jobs that are completed within a time window around the due date do not incur any penalties. There are many more variations of the nonregular measures of performance. Merten and Muller (1972), Eilon and Chowdhury (1977) and Bagchi et al. (1987b) consider the minimization of completion times variance about a common due date. Cheng (1984) has also studied the problem with different due dates but with the condition that the due date for each job is some common multiple of its processing time. Cheng (1987b) considers the problem of minimizing the maximum deviation of job completion time about a common due date. Other researchers such as Dessouky and Margenthaller (1972) and Lakshminarayan et al. (1978) have also considered the various forms of the problem. The first NP-Completeness proof for a nonregular measure of performance can be found in Garey et al. (1988).

Common Due Date One-Machine Earliness-Tardiness Problem with Unit Weights

The importance of common due date problems cannot be overemphasized since they frequently occur in many industrial settings. As has been mentioned, Bagchi (1989) provides four scenarios as to the practical existence of the common due date situations: in sub-assembly problems, in aggregate shipping, in kitting problems, and in multi-job customer orders.

The study of the one-machine scheduling problems is, by its own merit, important in the overall context of job shop scheduling. As was mentioned in Chapter 1, for one to study the complicated environment of job shop scheduling, one usually approaches the problem by decomposing the shop into a number of one-machine work centers. With this in mind, a thorough understanding of the one-machine work centers is crucial since any inaccuracies in the treatment of the one-machine work center problems will be magnified in the overall shop.

We define the following notation in addition to the ones defined in Chapter 1 (see Figure 3-1). Let

B = set of nontardy jobs, $\{j : C_j \leq D\}$

A = set of tardy jobs, $\{j : C_j > D\}$

$B_{[i]}$ = job processed in the i^{th} to last position of set B

$A_{1[i]}$ = job processed in the i^{th} position of set A_1

$A_{2[i]}$ = job processed in the i^{th} position of set A_2

Z_σ = total earliness-tardiness penalty for schedule σ

Note that the actual sequence of jobs will be BUA_1UA_2.

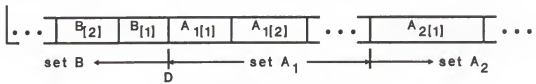


Figure 3-1. Schedule σ.

The first few articles on the problem (see for examples, Kanet, 1981; Sundararaghavan and Ahmed, 1984; Bagchi et al., 1986; Hall, 1986) assume a restrictively large common due date so as not to

constrain the schedule. This restriction is equivalent to the endogenous (internally determined) due date situation. Under this restriction, the problem can be easily solved in polynomial time. However, when the due date is assumed to be arbitrarily fixed (or assumed to be exogenous), the problem has been shown recently by Hall et al. (1989) to be NP-Complete. They have also provided a pseudo-polynomial dynamic programming algorithm of complexity $O(n\Sigma_j p_j)$ to solve the problem.

As can be seen from the complexity, the dynamic programming algorithm is highly dependent on the problem instance. Furthermore, in practical industrial settings, schedulers are not equipped to run complex algorithms. In such instances, quick heuristics that provide satisfactory solutions are preferable to sophisticated but slow optimal algorithms. From our numerous industrial contacts, we learn that there are strong demands which drive daily schedulers to use heuristics. Hence, the study of heuristics is justified.

Sundararaghavan and Ahmed (1984) provides the first such heuristic (S&A Algorithm). The S&A Algorithm assumes the first job to start at time zero. Since jobs are continuously processed, the last job, by the zero start time (ZST) assumption, will complete at time $MS = \Sigma_j p_j$. The S&A Algorithm then iteratively assigns jobs to be processed before or after the due date beginning with the first or the last position.

Baker and Chadowitz (1989) show that the zero start time assumption can result in poor schedules and that the algorithm can be

arbitrarily bad. They improve and extend the heuristic to include nonzero start time (B&C Algorithm). However, they do not provide an error bound on the B&C Algorithm. Instead, they empirically show that the B&C Algorithm performs better than S&A Algorithm. They also propose a neighborhood exchange procedure. Krieger and Raghavachari (1988) have also proposed a heuristic for the problem. They also do not provide any analytical results.

As far as we know, besides the three results mentioned above, little or no work has been done on analytical studies of error bounds of heuristics for the problem. This motivates us to develop a simple heuristic that obeys the V-shape property for which we can obtain a tight worst error bound. We will also provide a practical and systematic guideline for determining the performance of the heuristic.

Before we state the algorithm, we will reiterate the following three lemmas which are well known results (see for example Hall et al., 1989).

Lemma 3.1 (V-Shape Property): In an optimal schedule, the jobs that are completed on or before the due date are arranged in nonincreasing order of processing times (LPT) while the jobs that are started on or after the due date are arranged in nondecreasing order of processing times (SPT).

Lemma 3.2: Jobs in an optimal schedule are processed consecutively, without idle times.

Lemma 3.3: There exists an optimal schedule in which either the first job starts at time zero or there exists some job j such that $C_j = D$.

The heuristic we shall provide is motivated by the equivalence between the endogenous (restrictively large) common due date earliness-tardiness problem and the two-parallel machines problem of minimizing sum of job flow times as indicated by Bagchi et al. (1986) and Kubiak et al. (1990). We will now formally state the algorithm.

Algorithm 3.1

1. Alternately assign jobs beginning with job 1 to $B_{[1]}$ and job 2 to $A_{1[1]}$ if n is odd or job 1 to $A_{1[1]}$ and job 2 to $B_{[1]}$ if n is even until some job i cannot be assigned to set B (i.e., until the sum of processing times of jobs already assigned to set B before job i plus p_i is greater than D).
2. Transfer job p_{i-1} to last position of set B if possible and assign jobs i,i+1,...,n to set A_2. Otherwise, assign jobs i-1,i,i+1,...,n to set A_2. Calculate Z_σ for σ=BUA_1UA_2.
3. Let the first job of set A_2 be denoted by t. Set D' = D-p_t and remove job t from the job set. If D' > 0, then go to step 4. Otherwise, go to step 5.
4. Alternately assign the new job set beginning with job 1 assigned to $B'_{[1]}$ and job 2 assigned to $A_1'_{[1]}$ until no other job can be assigned to set B'.

5. Assign job t to the last position of set B' and the remaining jobs to set A_2'. Calculate $Z_{\sigma'}$ for σ=B'$\cup A_1' \cup A_2'$.

6. If $Z_{\sigma'} \geq Z_{\sigma}$, choose σ as the solution. Otherwise, choose σ'.

Remark: (1) Job $B_{[1]}$ always completes at time D and job $A_{1[1]}$ always starts at time D.

(2) Set A_2 can be viewed as the set of "left over" jobs that cannot be assigned to set B because of the small due date constraint.

(3) Sorting the jobs in nondecreasing order of processing times requires O(nlogn) and Algorithm 3.1 has a complexity of O(n). Hence the overall complexity is O(nlogn).

Example 1

D = 10.

Job : j	1	2	3	4	5	6
Processing time : p_j	2	4	5	8	11	12

Steps:

1 : n is even, $A_{[1]} = 1$, $\Sigma_{h\in B} p_h + p_2 = 4 < D$. Hence $B_{[1]} = 2$. $A_{[2]} = 3$, $\Sigma_{h\in B} p_h + p_4 = 12 > D$, job 4 cannot be assigned to set B.

2 : $p_3 = 5 < (D-\Sigma_{h\in B} p_h) = 6$. Hence $A_{1[2]} = 0$, $B_{[2]} = 3$, $A_{2[1]} = 4$, $A_{2[2]} = 5$, and $A_{2[3]} = 6$. $Z_{\sigma} = 70$ (see Figure 3-2).

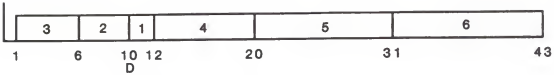


Figure 3-2. σ after step 2.

3 : $t = A_2[1] = 4$, $D' = D - p_4 = 10 - 8 = 2$, new job set = (1, 2, 3, 5, 6). $D' > 0$, go to step 4.

4 : $B'[1] = 1$, $A1'[1] = 2$.

5 : $B'[2] = 4$, $A_2'[1] = 3$, $A_2'[2] = 5$, and $A_2'[3] = 6$. $Z_\sigma' = 67$.

6 : $Z_\sigma' < Z_\sigma$. Hence, the solution is $\sigma' = (4, 1, 2, 3, 5, 6)$ with job 1 completing and job 2 starting at time D with total penalty of 67 (see Figure 3-3).

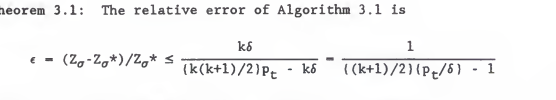


Figure 3-3. σ' after step 5.

For simplicity, we denote $|A_2|$ and $|A_2'|$ by k and k' respectively. We will now present the worst case analysis of Algorithm 3.1.

Theorem 3.1: The relative error of Algorithm 3.1 is

$$\epsilon = (Z_\sigma - Z_\sigma^*)/Z_\sigma^* \leq \frac{k\delta}{(k(k+1)/2)p_t - k\delta} = \frac{1}{((k+1)/2)(p_t/\delta) - 1}$$

where p_t is the processing time of the first job of set A_2, $\delta = D - \Sigma_{j \in B} p_j$, and Z_σ^* is the total penalty of the optimal solution.

Proof: Consider the schedule σ as obtained at the end of step 2 of Algorithm 3.1 and let $s = \Sigma_{h \in A_1} p_h$ (see Figure 3-4a). Let σ^* be the optimal schedule (see Figure 3-4b) and job j be the first job in σ^*. Note that the first job in σ is job (t-1).

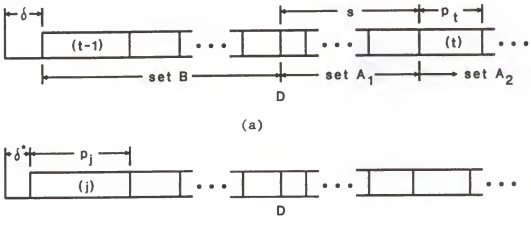


Figure 3-4. Given Schedules.
a) Schedule σ b) Schedule σ^*

Let X_σ* and X_σ be the sets of the first (t-1) jobs and let Y_σ* and Y_σ be the sets of the remaining jobs in schedules σ^* and σ respectively. Let $F_{ET}(V)$ be the earliness-tardiness penalty due to set V and $F_{FT}(W)$ be the minimum two-parallel machines sum of flow times of jobs in set W which is obtained using SPT. Note that $X_\sigma = (B \cup A_1)$, $Y_\sigma = A_2$, and X_σ contains the smallest (t-1) jobs. Hence,

$$F_{ET}(X_\sigma) = F_{FT}(X_\sigma \backslash (t\text{-}1)) \leq F_{ET}(X_\sigma^*) \tag{3.2}$$

Also, the completion time of the i^{th} job in set Y_σ is not more than δ time units later than that of the i^{th} job in set Y_σ*. Since sets Y_σ and Y_σ* contain k jobs, it can be easily shown that

$$F_{ET}(Y_\sigma) - F_{ET}(Y_\sigma^*) \leq k\delta \tag{3.3}$$

Combining equations (3.2) and (3.3), we obtain

$$Z_\sigma - Z_\sigma^* = [F_{ET}(X_\sigma)+F_{ET}(Y_\sigma)] - [F_{ET}(X_\sigma^*)+F_{ET}(Y_\sigma^*)]$$
$$\leq k\delta \qquad (3.4)$$

Furthermore, from Figure 3-4a, we can see that

$$Z_\sigma \geq \Sigma_{j\in Y_\sigma}(C_j - D)$$
$$\geq \sum_{i=1}^{k} (k-i+1)p_t$$
$$= (k(k+1)/2)p_t \qquad (3.5)$$

Hence, from equations (3.4) and (3.5),

$$Z_\sigma^* \geq Z_\sigma - k\delta$$
$$\geq (k(k+1)/2)p_t - k\delta \qquad (3.6)$$

Using equations (3.4) and (3.6), we finally obtain

$$\epsilon = (Z_\sigma - Z_\sigma^*)/Z_\sigma^*$$
$$\leq \frac{k\delta}{(k(k+1)/2)p_t - k\delta} \qquad (3.7)$$

This completes the proof of Theorem 3.1. Q.E.D.

Note that the proof of Theorem 3.1 does not use schedule σ' obtained from step 5 and that the relative error obtained thus far is rather loose. In fact, for $k = 1$, if p_t is very close to δ (note that $p_t > \delta$), then using equation (7), ϵ approaches ∞. Hence, Algorithm 3.1, without steps 3 to 5, can be arbitrarily bad. This prompts us to incorporate those steps into the algorithm. However, if k is large or if δ is small, the relative error is already very

small as the following table (Table 3-1) shows. Hence, under such conditions, steps 3 to 5 may not be necessary. Table 3-1 allows practitioners to make a systematic decision whether the additional steps are necessary.

Table 3-1. Upper Bounds on ϵ for ET Problem Given k and p_t/δ.

k	p_t/δ			
	2	4	6	8
1	1.00	0.33	0.20	0.14
2	0.50	0.20	0.13	0.09
3	0.33	0.14	0.09	0.07
4	0.25	0.11	0.07	0.05
5	0.20	0.07	0.06	0.04

Corollary 3.1. If $k = 0$ or $\delta = 0$, the sequence obtained from Algorithm 3.1 is optimal.

Proof: Directly obtainable from Theorem 3.1. Q.E.D.

We will next provide a better error bound on Algorithm 3.1 after steps 3 to 5 are implemented.

Theorem 3.2: The relative error of Algorithm 3.1 is

$$\epsilon = (Z_\sigma - Z_{\sigma^*})/Z_{\sigma^*} \leq 0.5$$

where σ^* is the optimal schedule. Furthermore, this bound is tight.

Proof: To analyze the actual worst case error bound of Algorithm 3.1, we will consider two cases: (1) $s \geq \delta$ and (2) $s < \delta$, where $s = \Sigma_{h\in A_1} p_h$ and $\delta = D - \Sigma_{h\in B} p_h$.

(1) $s \geq \delta$:

From Figure 3-4a, a tighter lower bound on Z_σ^* can be obtained as

$$Z_\sigma \geq (k(k+1)/2)p_t + (k+1)s \tag{3.8}$$

Again, from equation (3.4),

$$\begin{aligned} Z_\sigma^* &\geq Z_\sigma - k\delta \\ &\geq (k(k+1)/2)p_t + (k+1)s - k\delta \\ &> (k(k+1)/2)\delta + \delta \quad \text{since } s \geq \delta \text{ and } p_t > \delta \end{aligned} \tag{3.9}$$

Finally, from equations (3.4) and (3.9), we obtain

$$\begin{aligned} \epsilon &= (Z_\sigma - Z_\sigma^*)/Z_\sigma^* \\ &< \frac{k\delta}{(k(k+1)/2)\delta + \delta} \\ &\leq 0.5 \quad \text{for } k \geq 1 \end{aligned} \tag{3.10}$$

(2) $s < \delta$:

For this case, we will look at the cases $j \leq t\text{-}1$ and $j > t\text{-}1$ separately, where j is the first job in the optimal schedule σ^* and (t-1) is the first job in σ.

(2.1) $j \leq t\text{-}1$:

We will again compare schedules σ and σ^*. From Lemma 3.1 (V-shape Property), jobs (t, t+1, ..., n) are still processed last in σ^*. Let X_σ and X_{σ^*} be the set of the first t jobs in σ and σ^* respectively.

Let Y_σ and $Y_\sigma*$ be the remaining jobs in σ and σ^* respectively. Note that sets Y_σ and $Y_\sigma*$ each contains (k-1) jobs.

$$\begin{aligned} F_{ET}(X_\sigma) &= F_{FT}(X_\sigma/(t,t\text{-}1)) + (s+p_t) \\ &\le F_{FT}(X_\sigma/(t\text{-}1)) + s \quad \text{since } C_t \ge p_t \text{ in } F_{FT}(X_\sigma/(t\text{-}1)) \\ &\le F_{FT}(X_\sigma/(j)) + s \\ &\le F_{ET}(X_\sigma*) + s \end{aligned} \tag{3.11}$$

For jobs in set Y_σ and $Y_\sigma*$, the completion time of any job in Y_σ is not more than δ unit times later than that of the corresponding job in $Y_\sigma*$. Hence,

$$F_{ET}(Y_\sigma)\text{-}F_{ET}(Y_\sigma*) \le (k\text{-}1)\delta \tag{3.12}$$

Using equations (3.11) and (3.12), we obtain

$$Z_\sigma\text{-}Z_\sigma* \le (k\text{-}1)\delta + s \tag{3.13}$$

From Figure 3-4a and equation (3.13), we obtain

$$\begin{aligned} Z_\sigma* &\ge Z_\sigma - (k\text{-}1)\delta - s \\ &\ge (k(k+1)/2)p_t + (k+1)s - (k\text{-}1)\delta - s \\ &> (k(k+1)/2)\delta - (k\text{-}1)\delta + ks \quad \text{since } p_t > \delta \end{aligned} \tag{3.14}$$

Hence, combining equations (3.13) and (3.14), the following is obtained

$$\begin{aligned} \epsilon &= (Z_\sigma\text{-}Z_\sigma*)/Z_\sigma* \\ &< \frac{(k\text{-}1)\delta + s}{(k(k+1)/2)\delta - (k\text{-}1)\delta + ks} \end{aligned} \tag{3.15}$$

For k = 1,

$$\begin{aligned} \epsilon &< s/(\delta+s) \\ &< 0.5 \quad \text{since } \delta > s. \end{aligned}$$

For $k \geq 2$, let $a = (k-1)\delta$, $b = s$, $c = (k(k+1)/2)\delta - (k-1)\delta$, and $d = ks$. To prove $\epsilon < (a+b)/(c+d) \leq 0.5$, it suffices to prove $a/c \leq 0.5$ and $b/d \leq 0.5$:

$$a/c = \frac{(k-1)\delta}{(k(k+1)/2)\delta - (k-1)\delta} \leq 0.5 \quad \text{for } k \geq 2 \quad \text{and}$$

$$b/d = s/ks \leq 0.5 \quad \text{for } k \geq 2.$$

Hence, we obtain $\epsilon < 0.5$ for $k \geq 1$.

(2.2) $j > t-1$:

For this case, we will compare schedules σ' and σ^*. Let $s' = \Sigma_{h \in A_1'} p_h$, $\delta' = D - \Sigma_{h \in B'} p_h$, and let u be the first job of set A_2'. Note that job (u-2) is processed immediately after job t and job (u-1) is processed immediately before job u (see Figure 3-5). We will look at cases $s' \geq \delta'$ and $s' < \delta'$ separately.

(2.2.1): $s' \geq \delta'$

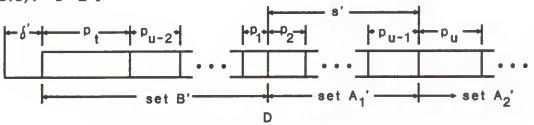


Figure 3-5. Schedule σ' with $s' \geq \delta'$.

Let X be the set of the first u jobs and Y be the set of the remaining jobs of schedule σ^*. Since $j > t-1$, the completion time of the i^{th} job of set A_2' is not more than δ' units of time later than the i^{th} job of set Y. Then

$$Z_{\sigma'} - Z_{\sigma^*} \leq k'\delta' \tag{3.16}$$

From Figure 3-5, we obtain

$$Z_\sigma^* \geq (k'(k'+1)/2)p_u + (k'+1)s' - k'\delta'$$
$$> (k'(k'+1)/2)\delta' + \delta' \quad \text{since } p_u > \delta' \text{ and } s' \geq \delta' \tag{3.17}$$

Hence

$$\epsilon < \frac{k'\delta'}{(k'(k'+1)/2)\delta' + \delta'}$$
$$\leq 0.5 \quad \text{for } k' \geq 1 \tag{3.18}$$

(2.2.2): $s' < \delta'$

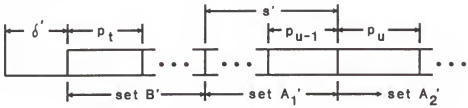


Figure 3-6. Schedule σ' with $s' < \delta'$.

Let $X_{\sigma'}$ and X_σ^* be the sets of the first (u+1) jobs in σ' and σ^* respectively. Let $Y_{\sigma'}$ and Y_σ^* be the sets of the remaining jobs in σ' and σ^*. Then

$$F_{ET}(X_{\sigma'}) = F_{FT}(\{1,\ldots,u-1\}) + (s'+p_u)$$
$$\leq F_{FT}(\{1,\ldots,u\}) + s'$$
$$\leq F_{ET}(X_\sigma^*) + s' \tag{3.19}$$

Furthermore, the completion time of the i^{th} job in set $Y_{\sigma'}$ is not more than δ' time units later than that of the i^{th} job in set Y_σ^* since $j > t-1$.

$$F_{ET}(Y_{\sigma'}) - F_{ET}(Y_\sigma^*) \leq (k'-1)\delta' \tag{3.20}$$

Hence, combining equations (3.19) and (3.20), we obtain

$$Z_{\sigma'} - Z_{\sigma}^{*} \leq (k'-1)\delta' + s' \qquad (3.21)$$

From Figure 3-6, similar to equation (3.5), we obtain

$$Z_{\sigma}^{*} \geq (k'(k'+1)/2)p_u + (k'+1)s' - (k'-1)\delta' - s'$$

$$> (k'(k'+1)/2)\delta' - (k'-1)\delta' + k's' \quad \text{since } p_u > \delta' \qquad (3.22)$$

Using equations (3.21) and (3.22), we finally obtain

$$\epsilon < \frac{(k'-1)\delta' + s'}{(k'(k'+1)/2)\delta' - (k'-1)\delta' + k's'} \qquad (3.23)$$

Similar to the proof of equation (3.15), we obtain $\epsilon < 0.5$ for $k' \geq 1$.

Thus, combining cases 1 and 2, we obtain $\epsilon \leq 0.5$. Q.E.D.

We will now show that this error is tight using the following example:

Example 2

Due date = 2M-1 where M is a large number.

Job : j	1	2	3
Processing time : p_j	M	M	M

Schedule σ obtained after step 2 of Algorithm 3.1:

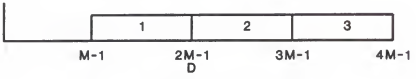


Figure 3-7. Schedule σ.

$Z_{\sigma} = 3M$

Schedule σ' obtained after step 5 of Algorithm 3.1:

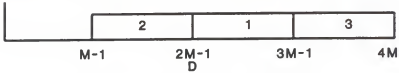


Figure 3-8. Schedule σ'.

$Z_{\sigma'} = Z_{\sigma} = 3M$

Optimal schedule:

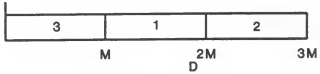


Figure 3-9. Schedule σ^*.

$Z_{\sigma}^* = (M-1) + 1 + (M+1) = 2M+1$

$\epsilon = (Z_{\sigma} - Z_{\sigma}^*)/Z_{\sigma}^* = (M-1)/(2M+1) \approx 0.5$ as M approaches infinity.

Hence, the error bound is attainable. Q.E.D.

However, as the following empirical analysis shows, the average error obtained is never more than 0.10. In the following section, we will provide the empirical results of the heuristic compared to the optimal solutions as would have been obtained using the dynamic programming algorithm provided by Hall et al. (1989).

Empirical Analysis

Each of the following problems (each row) is a replication of 100 randomly generated problems. The due date tightness is specified as the ratio of the due date to the sum of processing times. The smaller this value, the smaller the due date. Furthermore, for each

problem, the job processing times are randomly generated to follow a uniform distribution over (MinP,MaxP). The deviation is calculated as the difference of the heuristic solution to the optimal solution over the optimal solution. The deviations over all 100 randomly generated problems are averaged to give the entries of the last column. Tables 3-2, 3-3, 3-4, and 3-5 illustrate the results for the 10-, 20-, 30-, and 40-job problems.

The empirical results show that the maximum average deviation of the heuristic from the optimal solutions is less than 0.10. Averaging the entries of the fifth columns of the four tables, we obtain an overall average of 0.0241.

Table 3-2. 10-Job ET Problems.

Number of Jobs	Due Date Tightness	MinP	MaxP	Average Dev.
10	0.10	5	10	0.014
10	0.10	5	20	0.049
10	0.10	5	30	0.039
10	0.10	5	40	0.029
10	0.20	5	10	0.041
10	0.20	5	20	0.080
10	0.20	5	30	0.025
10	0.20	5	40	0.026
10	0.30	5	10	0.027
10	0.30	5	20	0.070
10	0.30	5	30	0.082
10	0.30	5	40	0.080
10	0.40	5	10	0.007
10	0.40	5	20	0.019
10	0.40	5	30	0.031
10	0.40	5	40	0.050

Table 3-3. 20-Job ET Problems.

Number of Jobs	Due Date Tightness	MinP	MaxP	Average Dev.
20	0.10	5	10	0.042
20	0.10	5	20	0.023
20	0.10	5	30	0.025
20	0.10	5	40	0.021
20	0.20	5	10	0.047
20	0.20	5	20	0.035
20	0.20	5	30	0.031
20	0.20	5	40	0.030
20	0.30	5	10	0.040
20	0.30	5	20	0.013
20	0.30	5	30	0.018
20	0.30	5	40	0.021
20	0.40	5	10	0.015
20	0.40	5	20	0.031
20	0.40	5	30	0.014
20	0.40	5	40	0.017

Table 3-4. 30-Job ET Problems.

Number of Jobs	Due Date Tightness	MinP	MaxP	Average Dev.
30	0.10	5	10	0.025
30	0.10	5	20	0.010
30	0.10	5	30	0.019
30	0.10	5	40	0.015
30	0.20	5	10	0.009
30	0.20	5	20	0.017
30	0.20	5	30	0.020
30	0.20	5	40	0.012
30	0.30	5	10	0.007
30	0.30	5	20	0.016
30	0.30	5	30	0.012
30	0.30	5	40	0.013
30	0.40	5	10	0.018
30	0.40	5	20	0.010
30	0.40	5	30	0.004
30	0.40	5	40	0.008

Table 3-5. 40-Job ET Problems.

Number of Jobs	Due Date Tightness	MinP	MaxP	Average Dev.
40	0.10	5	10	0.009
40	0.10	5	20	0.011
40	0.10	5	30	0.010
40	0.20	5	10	0.023
40	0.20	5	20	0.018
40	0.20	5	30	0.008
40	0.30	5	10	0.014
40	0.30	5	20	0.013
40	0.30	5	30	0.012
40	0.40	5	10	0.005
40	0.40	5	20	0.006
40	0.40	5	30	0.010

As a final note, for each σ (σ'), further improvements can be made by pairwise interchange between jobs in sets B and A (sets B' and A', respectively) as long as $\Sigma_{h\in B} P_h \leq D$ ($\Sigma_{h\in B'} P_h \leq D$, respectively). Furthermore, the schedule can be shifted to the left as long as the number of jobs in set A is greater than the number of jobs in set B, and the cost is reduced.

Chapter Summary

In this chapter, we have reviewed a number of literature on the nonregular performance measure. We then present some results for the common due date earliness-tardiness problems with unit weights. In particular, we propose a new heuristic that has a tight worst case error bound of 0.5 and an average error of less than 0.03. We have also provided a practical guideline as to when to use the dynamic programming algorithm.

CHAPTER 4
COMPARISON BETWEEN ROLLING HORIZON RELEASE POLICIES WITH AND WITHOUT PRIORITY TO PREVIOUSLY RELEASED JOBS

Introduction and Motivations

Rolling Horizon (RH), as we have mentioned in Chapter 2, is a job release policy. A job release policy is a form of input control, which dictates how newly arrived jobs should be released into the shop floor. It basically delays the processing of a new job by holding the job order as a paper work rather than as a work-in-process until certain shop conditions such as the beginning of a shift or the congestion level are satisfied.

One of the primary objectives for the use of input control is that it simplifies scheduling decisions by discretizing the decision epoches (time when a scheduling decision has to be made). That is, instead of making a scheduling decision every time a new job arrives (dynamic release policy), a shop supervisor only needs to do it once every time period. Input control is also used for smoothing production loads by delaying the entry of new jobs into the system during peak periods. A peak period is usually defined as the period during which time the system workload exceeds some threshold value.

From our industrial experience, in most practical instances, jobs are released into the floor every fixed interval of time (called a horizon), usually at the beginning of a workday or a shift. We

will call this type of release policies the "Rolling Horizon" policies of which we will distinguish two different policies.

In this chapter, we compare the performances of the two RH release policies using simulations. The two RH release policies are: (1) Rolling Horizon With Priority for Previously Released Jobs (RHP) and (2) Rolling Horizon Without Priority for Previously Released Jobs (RH). In particular, we will evaluate the performances and robustness of the two release policies for various shop conditions such as congestion levels and product mix. The different degree of congestions will be tested using varying "horizon tightness" which we will define later. The different product mix will be represented by varying the mean and the variance of job processing time. We will now formally define the two RH release policies studied.

Rolling Horizon with Priority for Previously Released Jobs (RHP)

Using this release policy, left over (unfinished) jobs from the previous horizons are given the first priority. That is, no new job can be processed while there are still unfinished jobs from the previous horizons, which have not been started. In effect, machines are not simultaneously available until all "old" jobs are completed. In Chapter 2 we have shown that under this assumption, the SPT algorithm yields an optimal solution with the minimum mean flow time.

Rolling Horizon without Priority for Previously Released Jobs (RH)

New jobs from current horizon are mixed with old jobs from the previous horizons without being distinguished. At the beginning of

the horizon, all jobs are treated equally and are processed in the SPT order. Any "left over" (unfinished) jobs are carried over to the next horizon when they are again mixed with any new jobs released at the beginning of that horizon.

Dynamic Release Policy (DRP)

The Dynamic release policy is a policy where a rescheduling is done each time a new job arrives. This type of release policy is intuitively better than RH or RHP. However, the amount of rescheduling that needs to be done may outweigh the benefits, especially in situations when new jobs arrive frequently or when the production process is very complicated. In most practical situations, new jobs are not even considered until the next time a shop floor schedule is being made. It is not our intention to show that this policy is better than the two RH release policies. Hence, we will only conduct the comparison between the two RH policies.

We will investigate the performance of the Shortest Processing Time algorithm (SPT) when applied to the two RH release policies. As we have mentioned in Chapter 2, this algorithm yields an optimal solution when machines are not all simultaneously available. This case, as previously described, may arise when there are unfinished jobs from the previous horizon and the machines are committed to finishing these jobs first (before processing the "new" jobs). This situation is depicted by the second release policy (RHP). We will next review some of the papers that exist on the subject of release

policies. As far as RH policies are concern, we have not been able to find any literature that specifically deals with the subject.

Literature Review

As we have previously mentioned, most of the realistic scheduling problems belong the class of NP-Complete problems in the strong sense. This makes the analytical study of scheduling problems difficult. Hence, many researchers have resorted to the use of simulation to test their proposed heuristics or their conjectures. One of the first papers on the application of simulation programs to scheduling problems is that by Eilon and Hodgson (1967). We shall not dwell any further on the use of simulation programs in scheduling.

One of the first papers on the use on input control in manufacturing environment is by Wight (1970). In that paper, he stresses the importance of controlling lead time defined as "that time that lapses between the moment it is determined that an item is needed and ordered to be replenished and that moment the item is available for use (p. 11)." This is achieved by establishing the system capacity and then releasing jobs at a planned rate and at the last possible moment. He treats the input control as a plan that needs to be made for the entire floor and for individual work centers.

Lankford (1982) also views input control as a tool for lead time management. He proposes the link between the individual work center input controls to the global Capacity Requirements Planning.

However, controlling individual work center is harder to achieve. Hence, input control cannot be effectively implemented in a job shop environment since it requires control of all work centers. According to him, it is sometimes more appropriate to control inputs at gateway (initial) work centers.

Bertrand (1983) studies the effect of workload information in a controlled release production system. In particular, a job is released into the shop floor once the workload falls below a specified value. This value is defined as the total amount of remaining processing times of all remaining operations in the shop, summed over all machines. He reports that the variance of lateness is reduced by using the time-phased workload information.

Baker (1984) investigates the advantages and disadvantages of remaining workload input control (similar to Bertrand, 1983) on a simple production environment. In particular, since input control may have conflicting effects on reducing scheduling complexity but at the same time reducing scheduling flexibility, its use may in fact be detrimental. He finds that input control maybe advantageous if used with dispatching rules that rely on shortest-first or critical ratio priorities. On the other hand, it does not improve the performance of the modified due date rule. The main conclusion is that input control maybe counterproductive.

Recently, Wein (1988) reports on favorable outcomes when input control is used in semiconductor wafer fabrication facility. He compares four input control policies: (1) Poisson input, (2)

deterministic input (constant job interarrival times), closed loop input (number of jobs in system is held constant), and (4) workload regulating input (jobs are released once the total amount of remaining work in the system for any bottleneck station falls below a prescribed level). He concludes that input controls provide bigger improvements on the performance of the wafer fabrication than lot sequencing. Furthermore, the improvements due to lot sequencing are highly dependent on the input control used and on the number of bottleneck stations.

In the next section, we will describe the simulation parameters that we use to compare the performances of the two release policies (RH and RHP).

Experimental Designs

We conduct a number of experiments by using a simulation program with varying parameters to reflect the various aspects of a shop floor such as congestion and product mix. The performance measures chosen are the mean flow time and the number of jobs completed within a number of horizons after their respective release times.

We define the flow time of a job as the length of time the job spends in the system from the release time (at the beginning of a particular horizon) until it finishes its last operation. This measure is chosen to test the performance of the SPT algorithm, which minimizes the mean flow time when all jobs are completed within the period they are released (if there is no left over job). The second

measure is chosen because of the fact that when RH is used, there is a possibility that long jobs will be continually delayed until all short jobs are finished resulting in a number of jobs being completed very late. Using RHP, on the other hand, may eliminate this problem since we continue processing the older jobs until all of them are finished before processing the new jobs.

We are looking at the scheduling of one work center shop. The work center, however, may contain more than one identical machines. Again, we assume no pre-emption. We also assume that a random number of jobs arrive at the beginning of a horizon and not continuously over time. This can be justified by the fact that jobs that arrive during a particular horizon will not be considered for processing until the beginning of the next horizon.

The number of jobs are randomly generated from a specific uniform distribution, which depends on the number of machines in the work center. This is purposely done so as to keep the overall work loads as similar as possible. In particular, the higher the number of machines, the more jobs are generated at the beginning of the shift (see Table 4-1). The mean number of jobs are calculated by:

Mean Number of Jobs = Number of Machines x 50

The length of horizon is also varied to account for "tight" to "loose" horizons. A horizon is "tight" if there are many jobs which are carried over to the next horizon(s) and is "loose" if there is only a few jobs (or no job) which are carried over. A "balanced"

horizon is defined as the horizon, which is equal to the sum of the mean of job processing times. A balanced horizon is calculated as:

Balanced Horizon = Mean Number of Jobs x Mean Processing Time

Any horizons smaller or larger than the balanced horizon are called "tight" or "loose" horizons respectively. A very tight and a tight horizons are 500 and 250 units of time less than the balanced horizon whereas a loose and a very loose horizons are 250 and 500 units of time more than the balanced horizon respectively. The different tightness levels are meant to simulate different levels of shop congestion. A tight horizon can be interpreted as analogous to a heavily congested floor. On the other hand, a loose horizon can be interpreted as a lightly congested floor.

Balanced Horizon = Mean Processing Time x Mean Number of Jobs
Very Tight Horizon = Balanced Horizon - 500
Tight Horizon = Balanced Horizon - 250
Loose Horizon = Balanced Horizon + 250
Very Loose Horizon = Balanced Horizon + 500

Job processing times are also randomly generated from uniform distributions with different means and variances. The experiment is designed to test the effects of the different means and variances of job processing time on the performances of the two rolling horizon release policies with respect to the two performance measures (the mean flow time and the number of jobs completed within a number of

horizons after release time). This is meant to differentiate between the different product mix. For instance, a large mean and a small variance in the processing times can be interpreted as a representation of a shop that makes only a few products but each product may take a long time to complete (e.g., aircraft manufacturing companies).

When studying the effects of increase in the mean of job processing time, we will keep the variance constant. On the other hand, when we want to isolate the effects of increase in the variance of job processing time, we keep the mean constant. The experimental designs for each of the two release policies (RHP and RH) are listed in the following table:

Table 4-1. Simulation Parameters.

Hor. Tight.	1 V.Tight	2 Tight	3 Balanced	4 Loose	5 V.Loose
# Mach # Jobs	1 U(30,70)	2 U(60,140)	3 U(90,210)	4 U(120,280)	5 U(150,350)
Proc.	U(0,60)	U(10,70)	U(20,80)	U(30,90)	U(40,100)
Time	U(40,60)	U(30,70)	U(20,80)	U(10,90)	U(0,100)

For each of the five different configuration of the work center (one to five machines), we conduct the study on the 10 different processing time distributions with the five different horizon tightness. Horizon tightness of one is for very tight horizon while tightness of five represents very loose horizon. The simulation is

then run for 15 horizons. The results obtained from the first and the last five horizons are discarded since the first five horizons are used to populate while the last five are used to flush out the system. The following observations are based on the five selected horizons.

Performance in Terms of the Mean Flow Time

We will now report on the results of the simulations conducted to investigate the performance of the different release polices on the mean flow time performance measure. As we have mentioned, we will only report the results obtained for the one-machine work center. Results for the other configurations of work center are found to be similar. We will first look at the results for the increase in the mean of job processing time and then review the results for the increase in the variance of job processing time.

Effects of Increase in the Mean of Job Processing Time

In this section we will thoroughly investigate the effects of an increase in the mean of job processing time. As we have mentioned, this is done to reflect the different nature of product mix manufactured by a particular company. For instance, a garment company will have a lower mean job processing time than a manufacturer of high technology equipment.

One of the intuitive observation is that for a given processing time distribution, the mean flow obtained using RHP is higher than that obtained using RH. For example, when $p_i \sim U(0,60)$, the mean

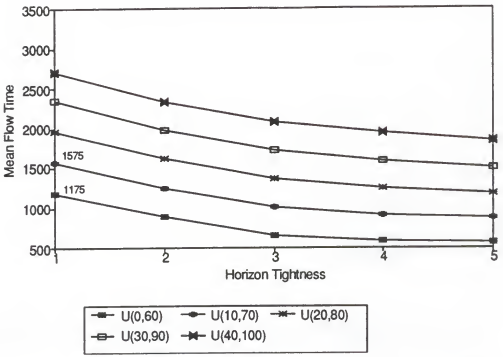
flow time obtained using RH when the horizon is very tight is 1175 while using RHP it is 1670 (see Figure 4-1).

This can be explained by the fact that in RHP, each of the new jobs that arrive at the beginning of the horizon has to wait for the amount of time the machines are tied up processing jobs from the previous horizons. Using RH, the SPT ordering, which is optimal for minimizing the mean flow time, is always maintained.

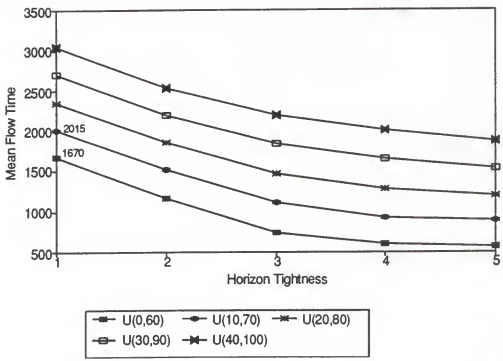
However, as the horizon tightness is reduced, for a given work center configuration, these differences diminish. In fact, when the horizon is very loose, the results are very close. For instance, when $P_i \sim U(0,60)$ with very loose horizon, the mean flow time for RH is 562 and for RHP is also 562 (see Figure 4-2). These results are consistent through out the processing time distributions (see Figure 4-3 for $P_i \sim U(40,100)$).

This is due to the fact that as the horizon becomes looser (shop is less congested), the number of left over jobs from previous horizons decreases. Hence, most of the new jobs do not have to wait to begin their processing.

Another observation that can be made is that as the mean of processing time is increased, the increase in the mean flow time using RHP is less than that using RH. For example (see Figure 4-1), as the mean is increased from 30 to 40 (from $P_i \sim U(0,60)$ to $P_i \sim U(10,70)$), the mean flow time for RH increases by 400 (from 1175 to 1575) while for RHP, it increases by only 345 (from 1670 to 2015).



(a)



(b)

Figure 4-1. Effects of Increase in the Mean of Job Processing Time on Mean Flow Time for One-Machine Work Center.
(a) RH Release Policy (b) RHP Release Policy

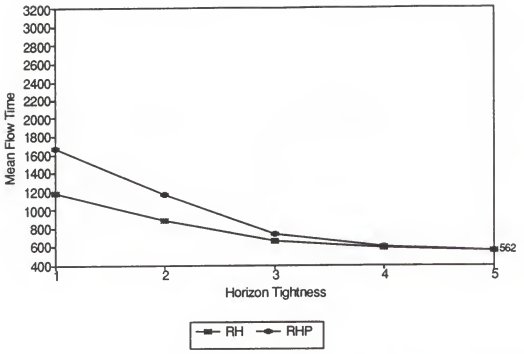


Figure 4-2. Comparison of Mean Flow Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(0,60)$.

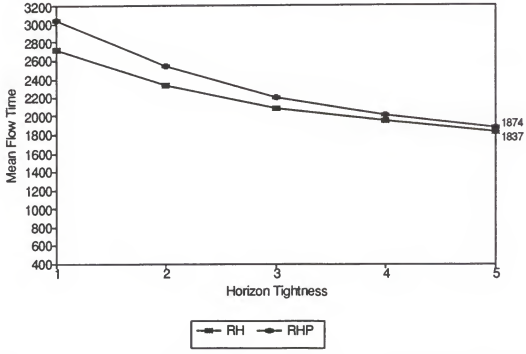


Figure 4-3. Comparison of Mean Flow Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(40,100)$.

This can be interpreted as RHP being slightly more robust than RH when there are differences in the mean of job processing time.

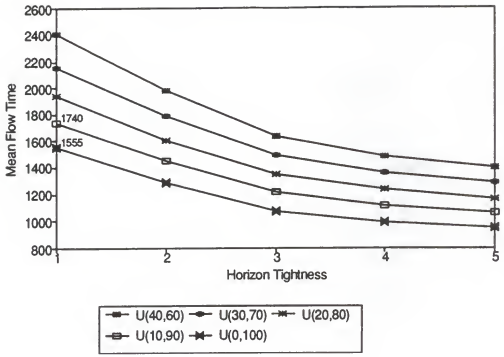
Effects of Increase in the Variance of Job Processing Time

In this section we will study the effects of an increase in the variance of job processing time. The mean is held constant at 50 time units. This is also done to reflect the different nature of product mix. For instance, a job shop facility will have a high degree of processing time variance.

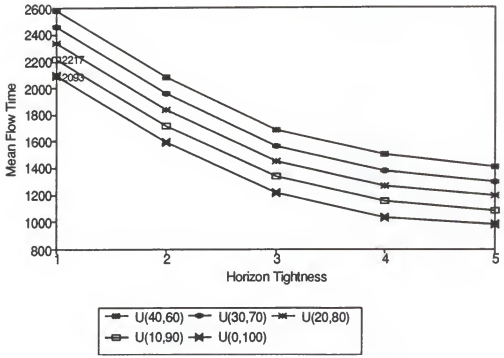
The most obvious effect is the decrease in the mean flow time as the variance of job processing time is increased (see Figure 4-4). For instance, for tight horizon, when $P_i \sim U(10,90)$, the mean flow time for RH is 1740 and for RHP is 2217. However, for $P_i \sim U(0,100)$, the mean flow time for RH decreases to 1555 and for RHP to 2093. Furthermore, as the horizon becomes tighter, the mean flow time increases at an increasing rate (see Figure 4-4), especially for RH release policy.

In all, RH still dominates RHP. However, the differences are less severe than that caused by the increase in the mean of job processing time, especially for small variance. From Figure 4-4, when $P_i \sim U(40,60)$, for tight horizon, the mean flow time for RH is larger than RHP by only 170 (2579 compared to 2409). But, when $P_i \sim U(0,100)$, the difference is 538 (2093 compared to 1555).

Similar to the effects caused by the increase in the mean flow time, these effects diminish as the horizon becomes looser, especially for RH policy. For example, when the horizon is very



(a)



(b)

Figure 4-4. Effects of Increase in the Variance of Job Processing Time on Mean Flow Time for One-Machine Work Center.
(a) RH Release Policy (b) RHP Release Policy

loose, for $P_i \sim U(40,60)$, the mean flow time for RH is 1404 and for RHP is 1413 (see Figure 4-5). Similarly, for the case when $P_i \sim U(0,100)$, the mean flow time for RH is 957 and for RHP is 978 (see Figure 4-6).

Again, we observe the fact that RHP is more robust than RH in the sense that the effects of increasing the variance of job processing time are less severe on the RHP policy than on the RH policy. In this case, the robustness is more pronounced than that due to the increase in the mean of job processing time (see Figure 4-4). For example, for tight horizon, when $P_i \sim U(10,90)$ is changed to $P_i \sim U(0,100)$, the mean flow time for RH decreases by 185 from 1740 to 1555 while for RHP it decreases by only 124 from 2217 to 2093.

Performance in Terms of the Number of Jobs Completed within a Certain Number of Horizons after the Release Time

For this performance measure, again we will only report on the results obtain for the one-machine work center shop. Similar results and conclusions are obtained and can be drawn for the different work center configurations. We will summarize the results of the simulations conducted to investigate the performance of the different release polices on the number of jobs completed within a certain number of horizons after their respective release times. We will first look at the results for the increase in the mean of job processing time and then review the results for the increase in the variance of job processing time.

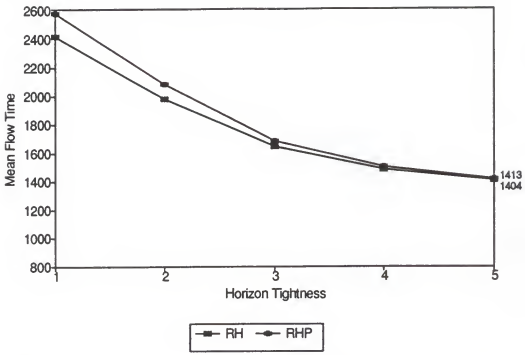


Figure 4-5. Comparison of Mean Flow Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(40,60)$.

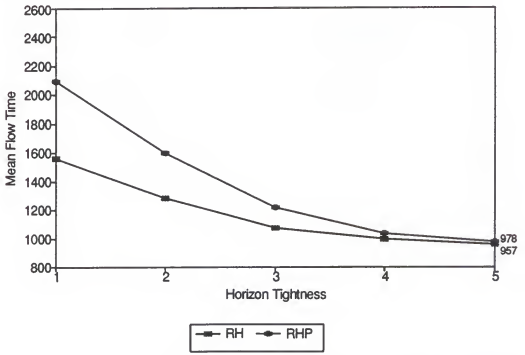


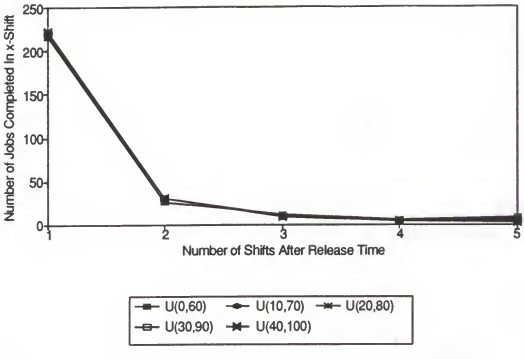
Figure 4-6. Comparison of Mean Flow Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(0,100)$.

Effects of Increase in the Mean of Job Processing Time

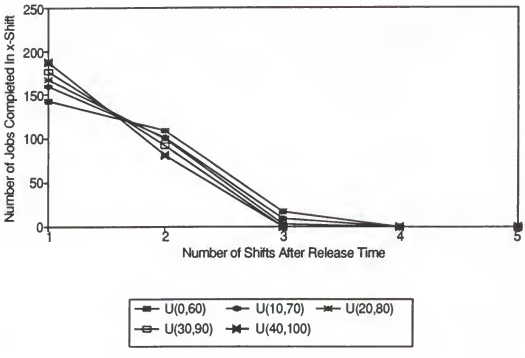
As expected, RH resulted in higher number of jobs that are completed closer to the release time since old jobs are not given the priority. However, it also resulted in more jobs that are completed far from their release times (very late jobs). On the other hand, RHP resulted in lower number of jobs that are completed very close to the release time but also lower number of jobs that are very late.

For a given horizon tightness, the increase in the mean of job processing time has more impact on RHP than on RH (RHP is more robust). This is especially true for very tight horizon (see Figure 4-7). In fact, both RH and RHP are almost identical for very loose horizon (see Figure 4-8). For tight horizon, using RHP, the larger the mean, the more products are completed within one horizon of the release times. However, the number of products completed within subsequent number of horizons decreases as the mean increases (see Figure 4-7(b)). On the other hand, for RH, the increase in the mean processing time does not have any significant impact (see Figure 4-7(a)).

The effects of horizon tightness can be better observed by looking at one particular processing time distribution. In particular, Figures 4-9 and 4-10 are for $P_i \sim U(0,60)$. For very tight horizon (see Figure 4-9), there is a significant difference between RHP and RH. Using RH, there are more jobs that are completed four or five horizons after they are released compared to RHP.

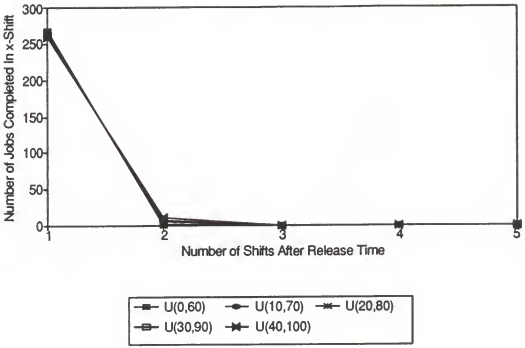


(a)

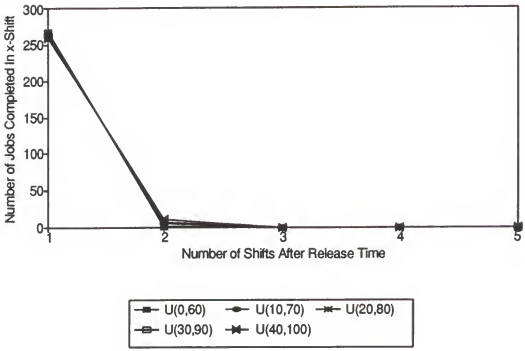


(b)

Figure 4-7. Effects of Increase in the Mean of Job Processing Time on Number of Jobs Completed Within a Number of Horizons After Release Time for One-Machine Work Center and Very Tight Horizon. (a) RH Release Policy (b) RHP Release Policy



(a)



(b)

Figure 4-8. Effects of Increase in the Mean of Job Processing Time on Number of Jobs Completed Within a Number of Horizons After Release Time for One-Machine Work Center and Very Loose Horizon. (a) RH Release Policy (b) RHP Release Policy

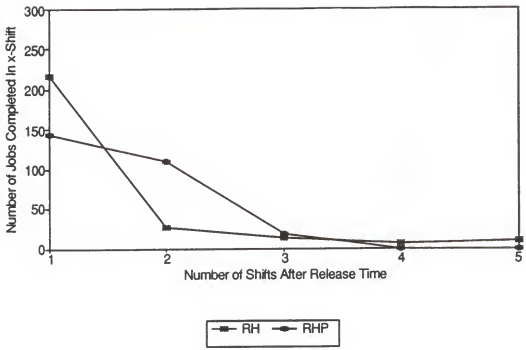


Figure 4-9. Comparison of Number of Jobs Completed Within a Number of Horizons After Release Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(0,60)$ with Very Tight Horizon.

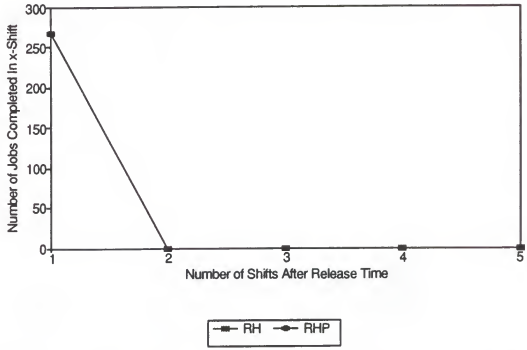


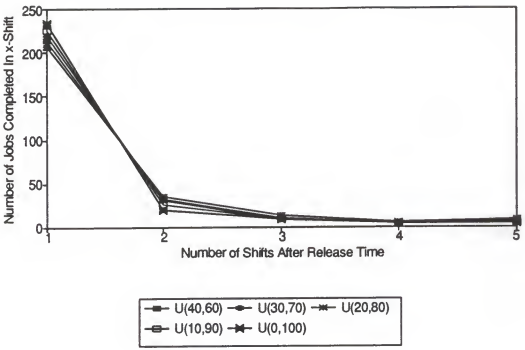
Figure 4-10. Comparison of Number of Jobs Completed Within a Number of Horizons After Release Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(0,60)$ with Very Loose Horizon.

However, as the horizon becomes looser (see Figure 4-10 for very loose horizon), both RH and RHP are identical. Again, this is because for very loose horizons, almost all jobs are completed within one horizon after being released since there is no congestion in the shop.

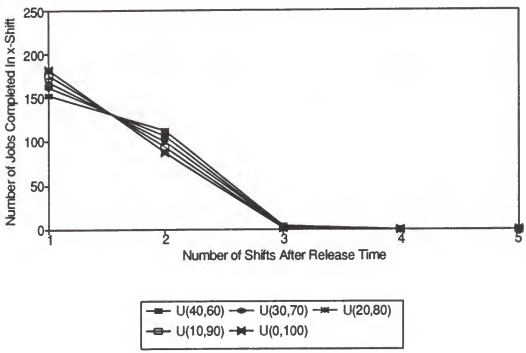
Effects of Increase in the Variance of Job Processing Time

Identical observations and conclusions as the effects of increase in the mean of job processing time can be made. The impact of increasing the variance of job processing time is greater for RHP release policy compared for RH, particularly for very tight horizon (see Figure 4-11). This impact, however, is not as pronounced as the impact caused by increasing the mean of job processing time. As we can see, this increase in the variance has also affected the performance measure using RH. Hence, RH policy, in this case is slightly more robust than RHP. However, as the horizon becomes looser, this impact diminishes (see Figure 4-12 for very loose horizon).

Again, the effects due to horizon tightness is better observed by looking at one processing time distribution, $P_i \sim U(40,60)$. From Figure 4-13, which illustrates the case of tight horizon, we can see that using RHP, there are more jobs completed before four horizons after being released. On the other hand, for very loose horizon (see Figure 4-14), RHP is the same as RHP.

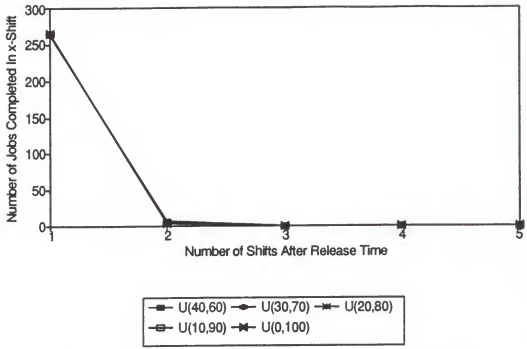


(a)

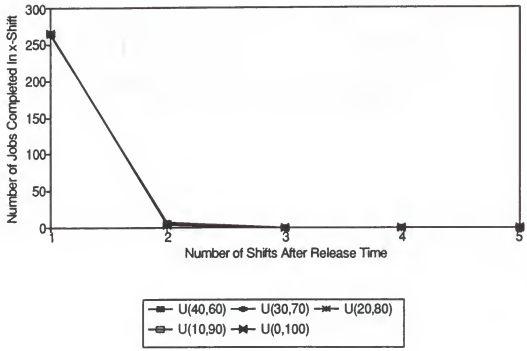


(b)

Figure 4-11. Effects of Increase in the Variance of Job Processing Time on Number of Jobs Completed Within a Number of Horizons After Release Time for One-Machine Work Center and Very Tight Horizon. (a) RH Release Policy (b) RHP Release Policy



(a)



(b)

Figure 4-12. Effects of Increase in the Variance of Job Processing Time on Number of Jobs Completed Within a Number of Horizons After Release Time for One-Machine Work Center and Very Loose Horizon. (a) RH Release Policy (b) RHP Release Policy

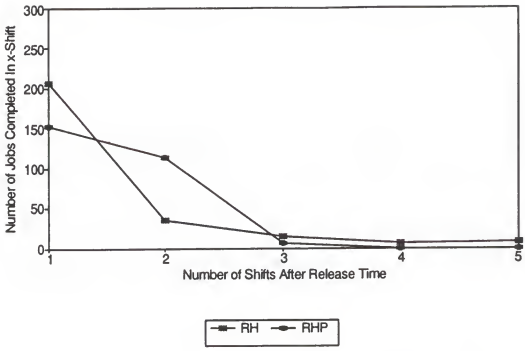


Figure 4-13. Comparison of Number of Jobs Completed Within a Number of Horizons After Release Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(40,60)$ with Very Tight Horizon.

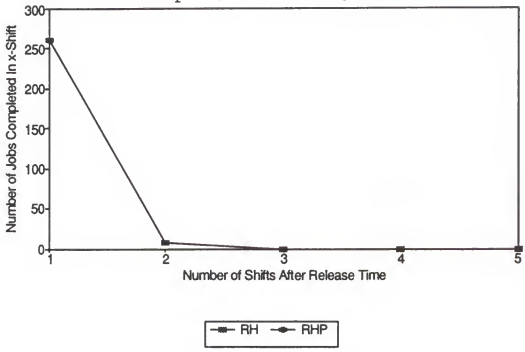


Figure 4-14. Comparison of Number of Jobs Completed Within a Number of Horizons After Release Time Obtained using RH and RHP for One-Machine Work Center for $P_i \sim U(40,60)$ with Very Loose Horizon.

Chapter Summary

From the various simulations that we have conducted, two general and rather intuitive conclusions can be drawn. First, RH is better than RHP especially for a heavily congested floor in terms of the mean flow time. Second, RHP is better than RH in the sense that the former completes more jobs sooner than the latter.

In terms of the minimization of the mean flow time, RHP is found to be slightly more robust than RH in that the effects of the variations in product mix have less severe effects on RHP than on RH. However, RH is more robust when it comes to maximizing the number of jobs completed within a certain number of horizons from the release times.

As to which one is more suitable for a particular company, the answer depends on which is more important to the company: maximizing throughput or maximizing number of jobs that are completed as soon as they are released. It also depends on the congestion level and the product mix peculiar to the company. The results obtained here can only serve as guidelines to help the company makes its decisions. The following two tables summarize the observations made for the purpose of comparing the RH release policy and the RHP release policy.

Table 4.2. Summary of Results for Mean Flow Time.

Effects of Increase in	Observations
Mean of Job Proc. Time	• Overall inc. in mean flow time • Inc. in mean flow time pretty consistent • Effects diminish as horizon becomes looser • RHP is slightly more robust than RH
Variance of Job Proc. Time	• Overall dec. in mean flow time • Inc. in mean flow time less for lower variance • Effects diminish as horizon becomes looser • RHP is more robust than RH

Table 4.3. Summary of Results for Number of Jobs Completed Within a Number of Horizons After Release Time.

Effects of Increase in	Observations
Mean of Job Proc. Time	• Overall, number of jobs completed within a certain number of horizons is higher for RHP • RH is more robust than RHP • Effects diminish as horizon becomes looser
Variance of Job Proc. Time	• Overall, number of jobs completed within a certain number of horizons is higher for RHP • RH is slightly more robust than RH • Effects diminish as horizon becomes looser

CHAPTER 5
SUMMARY AND CONCLUSIONS

In this chapter, we will summarize the results obtained in Chapters 2, 3, and 4 and provide some possible future directions for research.

Scheduling with Restricted Machine Availability

In Chapter 2, we study the problems of minimizing the sum of job flow times. We have shown that the SPT algorithm yields an optimal schedule for the problem of nonsimultaneous machine available times. In this case, we assume that machines are not all simultaneously available at the beginning of a work day due to prior commitments to jobs that have not been previously finished.

We have obtained results for the two-parallel machines problem whereby Machine 2 has prior commitments and is only available for a specified period of time while Machine 1 is assumed to be continuously available. We have proven that the recognition problem is NP-Complete and have proposed a pseudo-polynomial dynamic programming algorithm of complexity $O(nR_2)$ where R_2 is the length of time Machine 2 is available. We also analyze the robustness of the SPT algorithm and show that it has a tight worst case error bound of 0.5. However, we empirically show that the overall average error is not more than 0.03.

Furthermore, we have also obtained some results for the scheduling problem with preventive maintenance. The assumption is that the machines are taken out of production at specified times for preventive maintenance. The length of maintenance is assumed to be a constant and that the maintenance is done on a rotating basis. For the single-machine problem, we provide a shorter NP-Completeness proof than the one propose by Adiri et al. (1989). We have also shown that the worst case error bound they obtain is erroneous and have proven the correct tight worst case error to be 2/7. For the two-parallel machine problem, we propose a pseudo-polynomial dynamic programming algorithm of complexity $O(nR_1R_2)$, where R_1 and R_2 are the maintenance time for Machines 1 and 2 respectively, to solve the problem optimally.

Directions for future research lie in the extension of these initial results to the more general m machines problems. The final objective is to incorporate these different scenarios in the context of general job shop scheduling problem to make the problem more realistic.

Scheduling with Nonregular Measure of Performance

For the nonregular measure of performance of minimizing weighted sum of earliness and tardiness penalties, we investigate the case of exogenous common due date. We have provided the first heuristic for the exogenous common due date with unit weights problems for which a tight relative worst case error bound of 0.5 is obtained. This algorithm is based on the simple SPT algorithm and

has high potentials for improvement. Empirical results also indicate that the overall average error of the heuristic is less than 0.03.

In terms of future directions, we plan to further improve the performance of the above mentioned heuristic. We also intend to extend the initial results to obtain a heuristic for the general weight case. Other extensions, such as different due dates and multi-machine problems, can also be investigated. Another possible extension to the research would be to place the overall scenario in a JIT driven job shop to evaluate its true merits and potentials.

Comparison Between Rolling Horizon Release Policies with and without Priority to Previously Released Jobs

We have compared the two release policies and find that each of them has its own advantages and disadvantages. In particular, RH is better in minimizing the mean flow time while RHP is better in maximizing the number of jobs completed within a certain number of horizons after their respective release times. RHP is also found to be more robust than RH in that the effects of the variations in product mix have less severe effects on RHP than on RH. The results obtained can be used as guidelines in deciding which release policy is best for a particular company.

Future directions include extending the results to the general multi work center job shop. We could also test the performance of these release policies using different dispatching rules, other than the SPT. The interactions between input control and dispatching

rules are very complicated. There are still some areas which have not been touched by researchers mainly due to their complexities.

REFERENCES

Adiri, I., J. Bruno, E. Frostig, and A.H.G. Rinnooy Kan, "Single Machine Flow-Time Scheduling With a Single Breakdown," to appear in Acta Informatica, (1989).

Bagchi, U., "Due Date or Deadline Assignment to Multi-Job Customer Orders," Research Report, (1989), Department of Management, University of Texas, Austin, Texas.

Bagchi, U., Y.L. Chang and R.S. Sullivan, "Minimizing Absolute and Squared Deviations of Completion Times with Different Earliness and Tardiness Penalties and a Common Due Date," Naval Research Logistics, 34, (1987), pp. 739-751.

Bagchi, U., R.S. Sullivan and Y.L. Chang, "Minimizing Mean Absolute Deviation of Completion Times about a Common Due Date," Naval Research Logistics Quarterly, 33, (1986), pp. 227-240.

Bagchi, U., R.S. Sullivan and Y.L. Chang, "Minimizing Mean Squared Deviation of Completion Times about a Common Due Date," Management Science, 33, (1987), pp. 894-906.

Baker, K.R., Introduction to Sequencing and Scheduling, (1974), John Wiley and Sons, Inc., New York.

Baker, K.R., "The Effects of Input Control In a Simple Scheduling Model," Journal of Operations Management, 4, (1984), pp. 99-112.

Baker, K.R. and A.J. Chadowitz, "Algorithms for Minimizing Earliness and Tardiness Penalty with a Common Due Date," Working Paper No. 240, (1989), The Amos Tuck School of Business Administration, Dartmouth College, New Hampshire.

Baker, K.R. and G.D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review," Operations Research, 38, (1990), pp. 22-36.

Bector, C.R., Y.P. Gupta and M.C. Gupta, "Determination of an Optimal Common Due Date and Optimal Sequence in a Single Machine Job Shop," International Journal of Production Research, 26, (1988), pp. 613-628.

Bertrand, J.W.M., "The Use of Workload Information to Control Job Lateness in Controlled and Uncontrolled Release Production Systems," Journal of Operations Management, 3, (1983), pp. 79-92.

Blazewicz, J., G. Finke, R. Haupt, and G. Schmidt, "New Trends in Machine Scheduling," European Journal of Operational Research, 37, (1988), pp. 303-317.

Buxey, G., "Production Scheduling: Practice and Theory," European Journal of Operational Research, 39, (1989), pp. 17-31.

Chand, S. and D. Chhajed, "A Single-Machine Model for Determination of Optimal Due Dates and Sequence," Research Report, (1989), Krannert Graduate School of Management, Purdue University, West Lafayette, Indiana.

Cheng, T.C.E., "Optimal Due-Date Determination and Sequencing of n Jobs on a Single Machine," Journal of the Operational Research Society, 35, (1984), pp. 433-437.

Cheng, T.C.E., "A Duality Approach to Optimal Due-Date Determination," Engineering Optimization, 9, (1985), pp. 127-130.

Cheng, T.C.E., "A Note on the Common Due-Date Assignment Problem," Journal of the Operational Research Society, 37, (1986), pp. 1089-1091.

Cheng, T.C.E., "Minimizing the Average Deviation of Job Completion Times about a Common Due-Date: An Extension," Mathematical Modelling, 9, (1987a), pp. 13-15.

Cheng, T.C.E., "Minimizing the Maximum Deviation of Job Completion Time about a Common Due Date," Computers and Mathematics with Applications, 14, (1987b), pp. 279-283.

Cheng, T.C.E., "An Algorithm for the CON Due-Date Determination and Sequencing Problem," Computers and Operations Research, 14, (1987c), pp. 537-542.

Cheng, T.C.E., "Optimal Constant Due-Date Assignment and Sequencing," International Journal of Systems Science, 19, (1988a), pp. 1351-1354.

Cheng, T.C.E., "Optimal Common Due Date With Limited Completion Time Deviation," Computers and Operations Research, 15, (1988b), pp. 91-96.

Cheng, T.C.E., "Dynamic Programming Approach to the Single-Machine Sequencing Problem with Different Due-Dates," Computers and Mathematics with Applications, 19, (1990a), pp. 1-7.

Cheng, T.C.E., "A Note on a Partial Search Algorithm for the Single-Machine Optimal Common Due-Date Assignment and Sequencing Problem," Computers and Operations Research, 17, (1990b), pp. 321-324.

Cheng, T.C.E. and M.C. Gupta, "Survey of Scheduling Research Involving Due Date Determination Decisions," European Journal of Operational Research, 38, (1989), pp. 156-166.

Cheng, T.C.E. and H.G. Kahlbacher, "An Algorithmic Approach to a Common Due-Date Scheduling Problem in a Single Machine Systems," Preprint No. 176, (1989), Fachbereich Mathematik, Universitat Kaiserslautern, Germany.

Coffman, E.G. Jr. (ed.), Computer and Job-Shop Scheduling, (1976), John Wiley & Sons, New York.

Conway, R.W., W.L. Maxwell and L.W. Miller, Theory of Scheduling, (1967), Addison-Wesley, Massachusetts.

Davis, J.S. and J.J. Kanet, "Single Machine Scheduling with Non-Regular Convex Completion Costs," Research Report, (1989), Department of Management, Clemson University, Clemson, South Carolina.

Day, J.E. and M.P. Hottenstein, "Review of Sequencing Research," Naval Research Logistics Quarterly, 17, (1970), pp. 11-39.

De, P., J. Ghosh and C. Wells, "Scheduling About a Common Due Date With Earliness and Tardiness Penalties," Working Paper, (1989), University of Dayton, Dayton, Ohio.

Dempster, M.A.H., J.K. Lenstra and A.H.G. Rinnooy Kan (eds.), Deterministic and Stochastic Scheduling, (1982), Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham England, July 6-17, 1981, D. Reidel Publishing Company, Dordrecht, Holland.

Dessouky, M.I. and C.R. Margenthaller, "The One-Machine Sequencing Problem with Early Starts and Due Dates," AIIE Transactions, 4, (1972), pp. 214-222.

Du, J. and J.Y.-T. Leung, "Minimizing Total Tardiness on One Machine is NP-hard," Mathematics of Operations Research, 15, (1990), pp. 483-495.

Eilon, S. and I.G. Chowdhury, "Minimizing Waiting Time Variance in the Single Machine Problem," Management Science, 23, (1977), pp. 567-575.

Eilon, S. and R.M. Hodgson, "Job Shops Scheduling with Due Dates," International Journal of Production Research, 6, (1967), pp. 1-13.

Emmons, H., "Scheduling to a Common Due Date on Parallel Uniform Processors," Naval Research Logistics, 34, (1987), pp. 803-810.

French, S., Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop, (1982), John Wiley and Sons, New York.

Fry, T.D., R.D. Amstrong and J.H. Blackstone, "Minimizing Weighted Absolute Deviation in Single Machine Scheduling," IIE Transactions, 19, (1987), pp. 445-450.

Garey, M.R. and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, (1979), Freeman, California.

Garey, M.R., R.E Tarjan and G.T. Wilfong, "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties," Mathematics of Operations Research, 13, (1988), pp. 330-348.

Graves, S.C., "A Review of Production Scheduling", Operations Research, 29 (1981), pp. 646-675.

Hall, N.G., "Single- and Multiple-Processor Models for Minimizing Completion Time Variance," Naval Research Logistics Quarterly, 33, (1986), pp. 49-54.

Hall, N.G., W. Kubiak and S.P. Sethi, "Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times about a Restrictive Common Due Date," to appear in Operations Research, (1989).

Hall, N.G. and M.E. Posner, "Earliness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times about a Common Due Date," to appear in Operations Research, (1989).

Hoogeveen, J.A., H. Oosterhout and S.L. van de Velde, "New Lower and Upper Bounds for Scheduling Around a Small Common Due Date," Report BS-R9030, (1990), Centre for Mathematics and Computer Science, Amsterdam, the Netherlands.

Hoogeveen, J.A. and S.L. van de Velde, "Scheduling Around a Small Common Due Date," Report BS-R8914, (1989), Centre for Mathematics and Computer Science, Amsterdam, the Netherlands.

Jackson, J.R., "Scheduling a Production Line to Minimize Maximum Tardiness," Research Report 43, (1955), Management Sciences Research Project, University of California at Los Angeles, Los Angeles, California.

Kahlbacher, H.G., "Scheduling With Monotonous Earliness and Tardiness Penalties," Preprint No. 156, (1989), Fachbereich Mathematik, Universitat Kaiserslautern, Germany.

Kanet, J.J., "Minimizing the Average Deviation of Job Completion Times about a Common Due Date," Naval Research Logistics Quarterly, **28**, (1981), pp. 643-651.

Kao, T.Y. and E.A. Elsayed, "Minimization of the Mean Flow Time of Jobs on Two Parallel Machines with a Machine Availability Constraint," Working Paper 89-104, (1989), Department of Industrial Engineering, Rutgers University, New Jersey.

Karp, R.M, "Reducibility among Combinatorial Problems," in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher (eds.), pp. 85-103, (1972), Plenum Press, New York.

Kawaguchi, T. and S. Kyan, "Deterministic Scheduling in Computer Systems: A Survey," Journal of the Operational Research Society of Japan, **31**, (1988), pp. 190-216.

Krieger, A.M. and M. Raghavachari, "Deterministic and Random Single Machine Scheduling with Due Dates and General Penalty Functions," Technical Report No. 37-88-163, (1988), Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, New York.

Kubiak, W., S. Lou and S. Sethi, "Equivalence of Mean Flow Time Problems and Mean Absolute Deviation Problems," Operations Research Letters, **9**, (1990), pp. 371-374.

Lageweg, B.J., E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Computer Aided Complexity Classification of Combinatorial Problems," Report BW 137/81, (1981a), Department of Operations Research, Stichting Mathematisch Centrum, Amsterdam, the Netherlands.

Lageweg, B.J., E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Computer Aided Complexity Classification of Deterministic Scheduling Problems," Report BW 138/81, (1981b), Department of Operations Research, Stichting Mathematisch Centrum, Amsterdam, the Netherlands.

Lakshminarayan, S., R. Lakshmanan, R.L. Papineau, and R. Rochette, "Optimal Single-machine Scheduling with Earliness and Tardiness Penalties," Operations Research, **26**, (1978), pp. 1079-1082.

Lankford, R., "Input/Output Control: Making It Work," APICS 25th Conference Proceedings, (1982), pp. 419-420.

Lawler, E.L., "A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness," Annals of Discrete Mathematics, 1, (1977), pp. 331-342.

Lawler, E.L., J.K. Lenstra and A.H.G. Rinnooy Kan, "Recent Developments in Deterministic Sequencing and Scheduling: A Survey," in Deterministic and Stochastic Scheduling, pp. 35-73, M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan (eds.), (1982), D. Reidel Publishing Company, Dordrecht, Holland.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity," Report # BS-R8909, (1989), Centre for Mathematics and Computer Science, Amsterdam, the Netherlands.

Lawler, E.L. and J.M. Moore, "A Functional Equation and Its Application to Resource Allocation and Sequencing Problems," Management Science, 16, (1969), pp. 77-84.

Lee, C.-Y., "Parallel Machines Scheduling with Nonsimultaneous Machine Available Time," Discrete Applied Mathematics, 30, (1991), pp. 53-61.

Lee, C.-Y., S.L. Danusaputro and C.-S. Lin, "Minimizing Weighted Number of Tardy Jobs and Weighted Earliness-Tardiness Penalties About a Common Due Date," Computers and Operations Research, 18, (1991), pp. 379-389.

Lenstra, J.K., Sequencing by Enumerative Methods, Mathematical Centre Tracts 69, (1977), Mathematisch Centrum, Amsterdam, Holland.

Lenstra, J.K. and A.H.G. Rinnooy Kan, "New Directions in Scheduling Theory," Operations Research Letters, 2, (1984), pp. 255-259.

Merten, A.G. and M.E. Muller, "Variance Minimization in Single Machine Sequencing Problems," Management Science, 18, (1972), pp. 518-528.

Ow, P.S. and T.E. Morton, "Filtered Beam Search In Scheduling," International Journal of Production Research, 26, (1988), pp. 35-62.

Ow, P.S. and T.E. Morton, "The Single Machine Early/Tardy Problem," Management Science, 35, (1989), pp. 177-191.

Panwalkar, S.S. and W. Iskander, "A Survey of Scheduling Rules," Operations Research, 25, (1977), pp. 45-61.

Panwalkar, S.S., M.L. Smith and A. Seidmann, "Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem," Operations Research, 30, (1982), pp. 391-399.

Quaddus, M.A., "A Generalized Model of Optimal Due-Date Assignment by Linear Programming," Journal of the Operational Research Society, 38, (1987), pp. 353-359.

Rachamadugu, R., "Dominant Solutions for the Early/Tardy Problem," Working Paper # 625, (1990a), School of Business Administration, University of Michigan, Ann Arbor, Michigan.

Rachamadugu, R., "Scheduling Jobs Against a Common Due Date with Proportionate Penalties," Working Paper # 631, (1990b), School of Business Administration, University of Michigan, Ann Arbor, Michigan.

Raghavachari, M., "A V-Shape Property of Optimal Schedule of Jobs About a Common Due Date," European Journal of Operational Research, 23, (1986), pp. 401-402.

Raghavachari, M., "Scheduling Problems with Non-Regular Penalty Functions - A Review," Opsearch, 25, (1988), pp. 144-164.

Raghavachari, M., "A Branch and Bound Algorithm for the Single Machine Absolute Deviation Problem with Different Earliness and Lateness Penalties," Technical Report No. 37-88-180, (1989), Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, New York.

Rinnooy Kan, A.H.G., Machine Scheduling Problems: Classification, Complexity and Computations, (1976), Nijhoff, The Hague, the Netherlands.

Rodamer, F.A. and K.P. White, Jr., "A Recent Survey of Production Scheduling," IEEE Transactions on Systems, Man, and Cybernetics, 18, (1988), pp. 841-851.

Seidman, A., S.S. Panwalkar and M.L. Smith, "Optimal Assignment of Due-Dates for a Single Processor Scheduling Problem," International Journal of Production Research, 19, (1981), pp. 393-399.

Seidman, A. and M.L. Smith, "Due Date Assignment for Production Systems," Management Science, 27, (1981), pp. 571-581.

Sen, T. and S.K. Gupta, "A State-of-Art Survey of Static Scheduling Research Involving Due Dates," Omega, 12, (1984), pp. 63-76.

Sidney, J.B., "Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties," Operations Research, 25, (1977), pp. 62-69.

Smith, W.E., "Various Optimizers for Single Stage Production," Naval Research Logistics Quarterly, 3, (1956), pp. 59-66.

Sundararaghavan, P.S. and M.U. Ahmed, "Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling," Naval Research Logistics Quarterly, 31, (1984), pp. 325-333.

Szwarc, W., "Single-Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due Date," Naval Research Logistics, 36, (1989), pp. 663-673.

Wein, L.M., "Scheduling Semiconductor Wafer Fabrication," IEEE Transactions on Semiconductor Manufacturing, 1, (1988), pp. 115-130.

Wight, O., "Input/Output Control: A Real Handle on Lead Time," Production and Inventory Management, 11, (1970), pp. 9-31.

BIOGRAPHICAL SKETCH

Surya Danusaputro Liman was born on December 4, 1962, in Surabaya, Indonesia. He is currently pursuing the degree of Doctor of Philosophy in the Department of Industrial and Systems Engineering at the University of Florida. He received his Master of Engineering degree in December, 1987, and his Bachelor of Science in Industrial Engineering degree in May, 1986, both from the University of Florida. He has published several articles in IIE Transactions, Computers and Industrial Engineering, Computers and Operations Research, and Microelectronics and Reliability. His primary research interest is in the area of production scheduling and inventory control. He is also a member of the Operations Research Society of America, the Institute of Management Sciences, the Institute of Industrial Engineers, Tau Beta Pi Engineering Honor Society, and Alpha Pi Mu Industrial Engineering Honor Society.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Chung-Yee Lee, Chair
Associate Professor of
Industrial and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

D.J. Elzinga
Professor and Chairman of
Industrial and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Louis A. Martin-Vega
Associate Professor of
Industrial and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Selcuk Erenguc
Professor of Decision and
Information Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1991

for Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School